xlsxwriter学习笔记

——杨双杰 2019年9月6日

workbook

.add_worksheet()/.add_format()

```
import xlsxwriter

workbook  = xlsxwriter.Workbook('filename.xlsx',
                                {'strings_to_numbers': True,
                                'strings_to_formulas': False,
                                'strings_to_urls': False,
                                'default_date_format': 'dd/mm/yy'})
worksheet1 = workbook.add_worksheet()           # Sheet1
worksheet2 = workbook.add_worksheet('Foglio2')  # Foglio2
worksheet3 = workbook.add_worksheet('Data')     # Data
worksheet4 = workbook.add_worksheet()           # Sheet4

format1 = workbook.add_format(props)  # Set properties at creation.
format2 = workbook.add_format()       # Set properties later.

worksheet1.write(0, 0, 'Hello Excel',format1)
worksheet2.write('A1', 'Hello world',format2)

workbook.close()
```

```
with xlsxwriter.Workbook('hello_world.xlsx') as workbook:
    worksheet = workbook.add_worksheet()

    worksheet.write('A1', 'Hello world')
```

.worksheets()/.get_worksheet_by_name()

```
workbook.sheets() # 返回所有worksheet对象
worksheet = workbook.get_worksheet_by_name('Sheet1')
```

.add_chart()/.add_chartsheet()

```
workbook.add_chart({'type': 'bar', 'subtype': 'stacked'})

#type:
#    area
```

```
#    bar
#    column
#    doughnut
#    line
#    pie
#    radar
#    scatter
#    stock

# subtype:
```

.set_size()

```
#设置工作簿窗口大小
workbook.set_size(1200, 800)
```

.set_tab_ratio()

```
workbook.set_tab_ratio(75) #水平滑块比例0-100
```

.set_properties()

```
workbook.set_properties({
    'title':    'This is an example spreadsheet',
    'subject':  'With document properties',
    'author':   'John McNamara',
    'manager':  'Dr. Heinz Doofenshmirtz',
    'company':  'of Wolves',
    'category': 'Example spreadsheets',
    'keywords': 'Sample, Example, Properties',
    'created':  datetime.date(2018, 1, 1),
    'comments': 'Created with Python and XlsxWriter'})
```

```
date = datetime.strptime('2016-12-12', '%Y-%m-%d')

workbook.set_custom_property('Checked by',        'Eve')
workbook.set_custom_property('Date completed',    date)
workbook.set_custom_property('Document number',   12345)
workbook.set_custom_property('Reference number', 1.2345)
workbook.set_custom_property('Has review',        True)
workbook.set_custom_property('Signed off',        False)
```

.define_name()

- 在工作簿中创建定义的名称以用作变量

```
workbook.define_name('Exchange_rate', '=0.96')
worksheet.write('B3', '=B2*Exchange_rate')
```

- 与在Excel中一样，这样定义的名称是工作簿的“全局”，可以从任何工作表中引用：

```
# Global workbook name.
workbook.define_name('Sales', '=Sheet1!$G$1:$H$10')
```

- 也可以使用以下语法在工作表名称前面加上工作表名称来定义工作表名称`'sheetname!definedname'`：

```
# Local worksheet name
workbook.define_name('Sheet2!Sales', '=Sheet2!$G$1:$G$10')
```

- 如果工作表名称包含空格或特殊字符，则必须遵循Excel约定并将其用单引号括起来：

```
workbook.define_name("'New Data'!Sales", '=Sheet2!$G$1:$G$10')
```

.add_vba_project()

- 用于使用从现有Excel xlsm文件中提取的二进制VBA项目文件将宏或函数添加到工作簿

```
workbook.add_vba_project('./vbaProject.bin')
```

.get_default_url_format()

- 获取未指定用户定义格式时使用的默认URL格式的副本`write_url()`

.workbook.set_calc_mode（）

- 设置工作簿的Excel计算模式。

  ```
  mode:
  * auto：默认值。当影响公式的公式或值发生更改时，Excel将重新计算公式。
  * manual：仅在用户需要时重新计算公式。一般按F9。
  * auto_except_tables：Excel将自动重新计算除表格以外的公式。
  ```

worksheet

.write()

```
worksheet.write(row,col,[number/string/format])
# 包含具体方法
# write_string()
# write_number()
# write_blank()
```

```
# write_formula()
# write_datetime()
# write_boolean()
# write_url()

worksheet.write(0, 0, 'Hello')          # write_string()
worksheet.write(1, 0, 'World')          # write_string()
worksheet.write(2, 0, 2)                # write_number()
worksheet.write(3, 0, 3.00001)          # write_number()
worksheet.write(4, 0, '=SIN(PI()/4)')   # write_formula()
worksheet.write(5, 0, '')               # write_blank()
worksheet.write(6, 0, None)             # write_blank()

cell_format = workbook.add_format({'bold': True, 'italic': True})
worksheet.write(0, 0, 'Hello', cell_format)  # Cell is bold and italic.
```

.write_formula()

```
worksheet.write_formula(0, 0, '=B3 + B4')
worksheet.write_formula(1, 0, '=SIN(PI()/4)')
worksheet.write_formula(2, 0, '=SUM(B1:B5)')
worksheet.write_formula('A4', '=IF(A3>1,"Yes", "No")')
worksheet.write_formula('A5', '=AVERAGE(1, 2, 3, 4)')
worksheet.write_formula('A6', '=DATEVALUE("1-Jan-2013")')
```

.write_array_formula()

- 对于返回值范围的数组公式，您必须指定返回值将写入的范围：

```
worksheet.write_array_formula(0, 0, 2, 0, '{=TREND(C1:C3,B1:B3)}')
worksheet.write_array_formula('A1:A3',    '{=TREND(C1:C3,B1:B3)}')
```

- 如果数组公式返回单个值，则 `first_` 和 `last_` 参数应该相同：

```
worksheet.write_array_formula('A1:A1', '{=SUM(B1:C1*B2:C2)}')
```

- 在这种情况下，只使用 `write_formula()` 或 `write()` 方法更容易：

```
# Same as above but more concise.
worksheet.write('A1', '{=SUM(B1:C1*B2:C2)}')
worksheet.write_formula('A1', '{=SUM(B1:C1*B2:C2)}')
```

.write_url()

- 支持两个本地URI：`internal:` 和 `external:`,用于内部工作表引用或外部工作簿和工作表引用。

```
# Link to a cell on the current worksheet.
worksheet.write_url('A1',  'internal:Sheet2!A1')
```

```
# Link to a cell on another worksheet.
worksheet.write_url('A2',  'internal:Sheet2!A1:B2')

# Worksheet names with spaces should be single quoted like in Excel.
worksheet.write_url('A3',  "internal:'Sales Data'!A1")

# Link to another Excel workbook.
worksheet.write_url('A4', r'external:c:\temp\foo.xlsx')

# Link to a worksheet cell in another workbook.
worksheet.write_url('A5', r'external:c:\foo.xlsx#Sheet2!A1')

# Link to a worksheet in another workbook with a relative link.
worksheet.write_url('A7', r'external:..\foo.xlsx#Sheet2!A1')

# Link to a worksheet in another workbook with a network link.
worksheet.write_url('A8', r'external:\\NET\share\foo.xlsx')
```

.write_comment()

```
worksheet.write('A1', 'Hello')
worksheet.write_comment('A1', 'This is a comment',{...})

{
    author
    visible
    x_scale
    width
    y_scale
    height
    color
    font_name
    font_size
    start_cell
    start_row
    start_col
    x_offset
    y_offset
}
```

.write_rich_string()

- 用于编写具有多种格式的字符串。例如——“This is **bold** and this is *italic* ”:

```
bold   = workbook.add_format({'bold': True})
italic = workbook.add_format({'italic': True})

worksheet.write_rich_string('A1',
                            'This is ',
                            bold, 'bold',
```

```
                            ' and this is ',
                            italic, 'italic')

# Some bold format and a default format.
bold    = workbook.add_format({'bold': True})
default = workbook.add_format()

# With default formatting:
worksheet.write_rich_string('A1',
                            'Some ',
                            bold, 'bold',
                            ' text')

# Or more explicitly:
worksheet.write_rich_string('A1',
                             default, 'Some ',
                             bold,    'bold',
                             default, ' text')
```

.write_row()

```
# Some sample data.
data = ('Foo', 'Bar', 'Baz')

# Write the data to a sequence of cells.
worksheet.write_row('A1', data)

# The above example is equivalent to:
worksheet.write('A1', data[0])
worksheet.write('B1', data[1])
worksheet.write('C1', data[2])
```

.write_column()

```
# Some sample data.
data = ('Foo', 'Bar', 'Baz')

# Write the data to a sequence of cells.
worksheet.write_column('A1', data)

# The above example is equivalent to:
worksheet.write('A1', data[0])
worksheet.write('A2', data[1])
worksheet.write('A3', data[2])
```

.set_row()

```
cell_format = workbook.add_format({'bold': True})

worksheet.set_row(row, height, cell_format,{'hidden':True, #隐藏
                                    'level':1, #大纲级别
                                    'collapsed':1}) #折叠
```

.set_column()

```
worksheet.set_column(0, 0, 20)   # Column  A   width set to 20.
worksheet.set_column(1, 3, 30)   # Columns B-D width set to 30.
worksheet.set_column('E:E', 20)  # Column  E   width set to 20.
worksheet.set_column('F:H', 30)  # Columns F-H width set to 30.

worksheet.set_column('B:G', None, None, {'hidden': 1, 'level': 1})
worksheet.set_column('H:H', None, None, {'collapsed': 1})
```

.insert_image()

```
worksheet.insert_image('B20',path,{
    'x_offset':        0,
    'y_offset':        0,
    'x_scale':         1,
    'y_scale':         1,
    'object_position': 2,
    'image_data':      None,
    'url':             None,
    'tip':             None,
})
```

.insert_chart()

```
chart = workbook.add_chart({type, 'column'})
worksheet.insert_chart('B5', chart,{
    'x_offset':        0,
    'y_offset':        0,
    'x_scale':         1,
    'y_scale':         1,
    'object_position': 1,
})
```

.insert_textbox()

```
worksheet.insert_textbox('B2', 'A simple textbox with some text')
# Size and position
width
height
x_scale
```

```
y_scale
x_offset
y_offset
object_position

# Formatting
line
border
fill
gradient
font
align
```

.insert_button()

- 此方法通常仅在与Workbook `add_vba_project()` 方法结合使用时将按钮绑定到嵌入式VBA项目中的宏时才有用:

```
# Add the VBA project binary.
workbook.add_vba_project('./vbaProject.bin')

# Add a button tied to a macro in the VBA project.
worksheet.insert_button('B3', {'macro':   'say_hello',
                               'caption': 'Press Me'})
# options
{
    'macro':    None,
    'caption':  'Button 1',
    'width':    64,
    'height':   20.
    'x_offset': 0,
    'y_offset': 0,
    'x_scale':  1,
    'y_scale':  1,
}
```

.data_validation()

```
worksheet.data_validation('B3', { 'validate': 'integer',
                                  'criteria': 'between',
                                  'minimum': 1,
                                  'maximum': 10,
                                  'ignore_blank': False,
                                  'input_title': 'Enter an integer:',
                                  'input_message': 'between 1 and 100',
                                  'error_title': 'Input value not valid!',
                                  'error_message': 'It should be an integer between 1 and
100'}) #类型限制

worksheet.data_validation('B13', {'validate': 'list',
```

```
                                    'source': ['open', 'high', 'close']}) #下拉列表

# Using 'value'.
worksheet.data_validation('A1', {'validate': 'integer',
                                 'criteria': 'greater than',
                                 'value': 100})

# Using 'minimum'.
worksheet.data_validation('B11', {'validate': 'decimal',
                                  'criteria': 'between',
                                  'minimum': 0.1,
                                  'maximum': 0.5})

# Using 'source'.
worksheet.data_validation('B10', {'validate': 'list',
                                  'source': '=$E$4:$G$4'})

# Using 'source' with a string list.
worksheet.data_validation('B13', {'validate': 'list',
                                  'source': ['open', 'high', 'close']})
```

```
validate:
    integer
    decimal
    list
    date
    time
    length
    custom
    any

criteria:
    between
    not between
    equal to                    ==
    not equal to                !=
    greater than                >
    less than                   <
    greater than or equal to    >=
    less than or equal to       <=
```

.conditional_format()

```
worksheet.conditional_format('B3:K12', {'type':     'cell',
                                        'criteria': '>=',
                                        'value':    50,
                                        'format':   format1})
```

.add_table()

```
worksheet.add_table('B3:F7', { ... })
```

.add_sparkline()

```
worksheet.add_sparkline(0, 5, {'range': 'A1:E1'})
worksheet.add_sparkline('F1', {'range': 'A1:E1'})
```

.write_comment()

```
worksheet.write_comment('C3', 'Hello', {'visible': False})
# options
{
'author': # worksheet.set_comments_author ()
"visible":
"x_scale":
"width":
"y_scale":
"height":
"color":
"font_name":
"font_size":
"start_cell":
"start_row":
"start_col":
"x_offset":
"y_offset":
}
```

.show_comments()

```
worksheet.show_comments()
```

.get_name()

```
for worksheet in workbook.worksheets():
    print worksheet.get_name()
```

.activate()

```
worksheet1 = workbook.add_worksheet()
worksheet2 = workbook.add_worksheet()
worksheet3 = workbook.add_worksheet()

worksheet3.activate()
```

.select()

```
worksheet.select() #选中工作表
```

.hide()

```
worksheet.hide() #隐藏工作表

worksheet.hide_zero() # 隐藏0值
```

.set_first_sheet()

```
worksheet.set_first_sheet() #将当前工作表设置为第一个可见工作表选项卡。
```

.merge_range()

```
merge_format = workbook.add_format({
    'bold':     True,
    'border':   6,
    'align':    'center',
    'valign':   'vcenter',
    'fg_color': '#D7E4BC',
})

worksheet.merge_range('B3:D4', 'Merged Cells', merge_format)
```

.autofilter()/.filter_column()/.filter_column_list()

```
worksheet.autofilter(0, 0, 10, 3)
worksheet.autofilter('A1:D11')

worksheet.filter_column('A', 'x > 2000')
worksheet.filter_column('B', 'x > 2000 and x < 5000')
# conditions
'x == Blanks'
'x == NonBlanks'
'x == b*'      # begins with b
'x != b*'      # doesn't begin with b
'x == *b'      # ends with b
```

```
'x != *b'      # doesn't end with b
'x == *b*'     # contains b
'x != *b*'     # doesn't contains b

worksheet.filter_column_list(2,   ['March', 'April', 'May'])
worksheet.filter_column_list('C', ['March', 'April', 'May'])
```

.set_selection()

```
worksheet1.set_selection(3, 3, 3, 3)  # 1. Cell D4.
worksheet2.set_selection(3, 3, 6, 6)  # 2. Cells D4 to G7.
worksheet3.set_selection(6, 6, 3, 3)  # 3. Cells G7 to D4.
worksheet4.set_selection('D4')        # Same as 1.
worksheet5.set_selection('D4:G7')     # Same as 2.
worksheet6.set_selection('G7:D4')     # Same as 3.
```

.freeze_panes()

```
worksheet.freeze_panes(1, 0)  # Freeze the first row.
worksheet.freeze_panes('A2')  # Same using A1 notation.
worksheet.freeze_panes(0, 1)  # Freeze the first column.
worksheet.freeze_panes('B1')  # Same using A1 notation.
worksheet.freeze_panes(1, 2)  # Freeze first row and first 2 columns.
worksheet.freeze_panes('C2')  # Same using A1 notation.

worksheet.freeze_panes(1, 0, 20, 0) #冻结第一行并使滚动区域从第二十行开始
```

.split_panes()

```
worksheet.split_panes(15, 0)     # First row.
worksheet.split_panes(0, 8.43)   # First column.
worksheet.split_panes(15, 8.43)  # First row and column.
```

.set_zoom()

```
worksheet.set_zoom() # 10-400
```

.right_to_left()

```
worksheet.right_to_left() # 从右到左 E D C B A
```

.set_tab_color()

- 设置工作表选项卡的颜色。

```
worksheet1.set_tab_color('red')
worksheet2.set_tab_color('#FF9900')  # Orange
```

.protect()

```
worksheet.protect('password',[...])

# Default values shown.
options = {
    'objects':               False,
    'scenarios':             False,
    'format_cells':          False,
    'format_columns':        False,
    'format_rows':           False,
    'insert_columns':        False,
    'insert_rows':           False,
    'insert_hyperlinks':     False,
    'delete_columns':        False,
    'delete_rows':           False,
    'select_locked_cells':   True,
    'sort':                  False,
    'autofilter':            False,
    'pivot_tables':          False,
    'select_unlocked_cells': True,
}

# 对于图表，允许的选项和默认值为
options = {
    'objects':               True,
    'content':               True,
}
```

.set_default_row()

```
worksheet.set_default_row(height,hide_unused_rows=True)
```

.outline_settings（）

- 用于控制Excel中轮廓的外观。

```
outline_settings (visible, symbols_below, symbols_right, auto_style )
'''
visible(bool) - 轮廓可见。可选，默认为True。
symbols_below (bool)  - 在轮廓栏下方显示行轮廓符号。可选，默认为True。
symbols_right (bool)  - 在轮廓条的右侧显示列轮廓符号。可选，默认为True。
auto_style (bool)  - 使用自动样式。可选，默认为False。
'''
```

.set_vba_name()

format

default format

```
cell_format = workbook.add_format() #default
cell_format.set_bold()
cell_format.set_font_color('red')

# equals
cell_format = workbook.add_format({'bold': True, 'font_color': 'red'})
```

modify format

```
cell_format = workbook.add_format({'bold': True, 'font_color': 'red'})
worksheet.write('A1', 'Cell A1', cell_format)

# Later...
cell_format.set_bold()
cell_format.set_italic()
cell_format.set_underline()
    # 1 = Single underline (the default)
    # 2 = Double underline
    # 33 = Single accounting underline
    # 34 = Double accounting underline

cell_format.set_font_strikeout() # 字体删除线
cell_format.set_font_script() # 1--上标；2--下标
cell_format.set_font_color('green')
cell_format.set_font_name('Times New Roman')
cell_format.set_font_size(30)
cell_format.set_num_format(index)
cell_format.set_locked(True) #锁定单元格
cell_format.set_hidden() #隐藏单元格公式，只显示结果
cell_format.set_align()
        # Horizontal alignment
            # left/center/right/fill/justify/center_across/distributed
       >>> cell_format.set_center_across () #相邻单元格居中对齐

        # Vertical alignment
            # top/vcenter/bottom/vjustify/vdistributed

cell_format.set_text_wrap() #单元格文本换行
cell_format.set_rotation(angle) #文本旋转
cell_format.set_reading_order() #1 left-to-right; 2 right-to-left
cell_format.set_indent(1) # 缩进级别 1, 2, ...
cell_format.set_shrink() #缩小文本以使其适合单元格
cell_format.set_pattern(1) # 设置背景图案0-18
```

```
cell_format.set_bg_color('green')# 设置背景颜色
cell_format.set_fg_color()

cell_format.set_border()
cell_format.set_left()
cell_format.set_right()
cell_format.set_top()
cell_format.set_bottom()

cell_format.set_border_color()
cell_format.set_left_color()
cell_format.set_right_color()
cell_format.set_top_color()
cell_format.set_bottom_color()

cell_format.set_diag_border()
cell_format.set_diag_color()
cell_format.set_diag_type()
    # 1 左下-右上
    # 2 左上-右下
    # 3 1+2

worksheet.write('B1', 'Cell B1', cell_format)
```

> set_num_format()

```
cell_format01.set_num_format('0.000')
worksheet.write(1, 0, 3.1415926, cell_format01)       # -> 3.142

cell_format02.set_num_format('#,##0')
worksheet.write(2, 0, 1234.56, cell_format02)         # -> 1,235

cell_format03.set_num_format('#,##0.00')
worksheet.write(3, 0, 1234.56, cell_format03)         # -> 1,234.56

cell_format04.set_num_format('0.00')
worksheet.write(4, 0, 49.99, cell_format04)           # -> 49.99

cell_format05.set_num_format('mm/dd/yy')
worksheet.write(5, 0, 36892.521, cell_format05)       # -> 01/01/01

cell_format06.set_num_format('mmm d yyyy')
worksheet.write(6, 0, 36892.521, cell_format06)       # -> Jan 1 2001

cell_format07.set_num_format('d mmmm yyyy')
worksheet.write(7, 0, 36892.521, cell_format07)       # -> 1 January 2001

cell_format08.set_num_format('dd/mm/yyyy hh:mm AM/PM')
worksheet.write(8, 0, 36892.521, cell_format08)       # -> 01/01/2001 12:30 AM

cell_format09.set_num_format('0 "dollar and" .00 "cents"')
worksheet.write(9, 0, 1.87, cell_format09)            # -> 1 dollar and .87 cents
```

```
# Conditional numerical formatting.
cell_format10.set_num_format('[Green]General;[Red]-General;General')
worksheet.write(10, 0, 123, cell_format10)  # > 0 Green
worksheet.write(11, 0, -45, cell_format10)  # < 0 Red
worksheet.write(12, 0,   0, cell_format10)  # = 0 Default color

# Zip code.
cell_format11.set_num_format('00000')
worksheet.write(13, 0, 1209, cell_format11)
```

> set_num_format(index)

Index	Index	Format String
0	0x00	`General`
1	0x01	`0`
2	0x02	`0.00`
3	0x03	`#,##0`
4	0x04	`#,##0.00`
5	0x05	`($#,##0_);($#,##0)`
6	0x06	`($#,##0_);[Red]($#,##0)`
7	0x07	`($#,##0.00_);($#,##0.00)`
8	0x08	`($#,##0.00_);[Red]($#,##0.00)`
9	0x09	`0%`
10	0x0a	`0.00%`
11	0x0b	`0.00E+00`
12	0x0c	`# ?/?`
13	0x0d	`# ??/??`
14	0x0e	`m/d/yy`
15	0x0f	`d-mmm-yy`
16	0x10	`d-mmm`
17	0x11	`mmm-yy`
18	0x12	`h:mm AM/PM`
19	0x13	`h:mm:ss AM/PM`
20	0x14	`h:mm`
21	0x15	`h:mm:ss`
22	0x16	`m/d/yy h:mm`
…	…	…
37	0x25	`(#,##0_);(#,##0)`
38	0x26	`(#,##0_);[Red](#,##0)`
39	0x27	`(#,##0.00_);(#,##0.00)`

Index	Index	Format String
40	0x28	`(#,##0.00_);[Red](#,##0.00)`
41	0x29	`_(* #,##0_);_(* (#,##0);_(* "-"_);_(@_)`
42	0x2a	`_($* #,##0_);_($* (#,##0);_($* "-"_);_(@_)`
43	0x2b	`_(* #,##0.00_);_(* (#,##0.00);_(* "-"??_);_(@_)`
44	0x2c	`_($* #,##0.00_);_($* (#,##0.00);_($* "-"??_);_(@_)`
45	0x2d	`mm:ss`
46	0x2e	`[h]:mm:ss`
47	0x2f	`mm:ss.0`
48	0x30	`##0.0E+0`
49	0x31	`@`

> set_border()

index	Name	weight	style
0	没有	0	
1	连续	1	`----------`
2	连续	2	`----------`
3	短跑	1	`- - - - - -`
4	点	1	`. . . . . .`
5	连续	3	`----------`
6	双	3	`==========`
7	连续	0	`----------`
8	短跑	2	`- - - - - -`
9	Dash Dot	1	`- . - . - .`
10	Dash Dot	2	`- . - . - .`
11	Dash Dot Dot	1	`- . . - . .`
12	Dash Dot Dot	2	`- . . - . .`
13	SlantDash Dot	2	`/ - . / - .`

pandas

Accessing XlsxWriter from Pandas

```
import pandas as pd

# Create a Pandas dataframe from the data.
df = pd.DataFrame({'Data': [10, 20, 30, 20, 15, 30, 45]})

# Create a Pandas Excel writer using XlsxWriter as the engine.
writer = pd.ExcelWriter('pandas_simple.xlsx', engine='xlsxwriter')

# Convert the dataframe to an XlsxWriter Excel object.
df.to_excel(writer, sheet_name='Sheet1')

# Get the xlsxwriter objects from the dataframe writer object.
workbook  = writer.book
worksheet = writer.sheets['Sheet1']
```

Adding Charts to Dataframe output

```
# Get the xlsxwriter objects from the dataframe writer object.
workbook  = writer.book
worksheet = writer.sheets['Sheet1']

# Create a chart object.
chart = workbook.add_chart({'type': 'column'})

# Configure the series of the chart from the dataframe data.
chart.add_series({'values': '=Sheet1!$B$2:$B$8'})

# Insert the chart into the worksheet.
worksheet.insert_chart('D2', chart)
```

Adding Conditional Formatting to Dataframe output

```
# Apply a conditional format to the cell range.
worksheet.conditional_format('B2:B8', {'type': '3_color_scale'})
```

Formatting of the Dataframe output

```
writer = pd.ExcelWriter("pandas_datetime.xlsx",
                        engine='xlsxwriter',
                        datetime_format='mmm d yyyy hh:mm:ss',
                        date_format='mmmm dd yyyy')
```

```
# Add some cell formats.
format1 = workbook.add_format({'num_format': '#,##0.00'})
format2 = workbook.add_format({'num_format': '0%'})

# Set the column width and format.
worksheet.set_column('B:B', 18, format1)

# Set the format but not the column width.
worksheet.set_column('C:C', None, format2)
```

Formatting of the Dataframe headers

```
# Turn off the default header and skip one row to allow us to insert a
# user defined header.
df.to_excel(writer, sheet_name='Sheet1', startrow=1, header=False)

# Get the xlsxwriter workbook and worksheet objects.
workbook  = writer.book
worksheet = writer.sheets['Sheet1']

# Add a header format.
header_format = workbook.add_format({
    'bold': True,
    'text_wrap': True,
    'valign': 'top',
    'fg_color': '#D7E4BC',
    'border': 1})

# Write the column headers with the defined format.
for col_num, value in enumerate(df.columns.values):
    worksheet.write(0, col_num + 1, value, header_format)
```

Handling multiple Pandas Dataframes

```
# Write each dataframe to a different worksheet.
df1.to_excel(writer, sheet_name='Sheet1')
df2.to_excel(writer, sheet_name='Sheet2')
df3.to_excel(writer, sheet_name='Sheet3')

# Position the dataframes in the worksheet.
df1.to_excel(writer, sheet_name='Sheet1')  # Default position, cell A1.
df2.to_excel(writer, sheet_name='Sheet1', startcol=3)
df3.to_excel(writer, sheet_name='Sheet1', startrow=6)

# Write the dataframe without the header and index.
df4.to_excel(writer, sheet_name='Sheet1',
             startrow=7, startcol=4, header=False, index=False)
```

Passing XlsxWriter constructor options to Pandas

```
writer = pd.ExcelWriter('pandas_example.xlsx',
                        engine='xlsxwriter',
                        options={'strings_to_urls': False})
```

chart

通过Workbook `add_chart()` 方法创建图表对象，其中指定了图表类型：

```
chart = workbook.add_chart({'type': 'column'})
```

然后使用 `insert_chart()` Worksheet方法将其作为嵌入式图表插入到工作 表中：

```
worksheet.insert_chart('A7', chart)
```

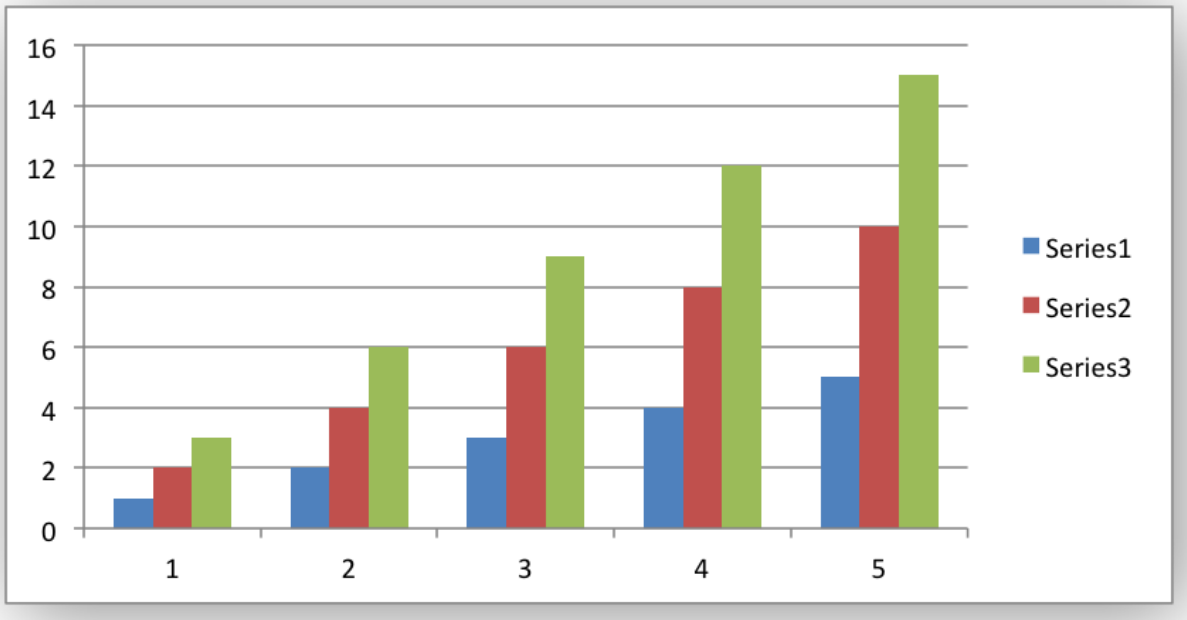
或者可以使用 `set_chart()` Chartsheet方法在图表中设置：

```
chartsheet = workbook.add_chartsheet()
# ...
chartsheet.set_chart(chart)
```

以下是一个小工作示例或添加嵌入式图表：

```
import xlsxwriter

workbook = xlsxwriter.Workbook('chart.xlsx')
worksheet = workbook.add_worksheet()

# Create a new Chart object.
chart = workbook.add_chart({'type': 'column'})

# Write some data to add to plot on the chart.
data = [
    [1, 2, 3, 4, 5],
    [2, 4, 6, 8, 10],
    [3, 6, 9, 12, 15],
]

worksheet.write_column('A1', data[0])
worksheet.write_column('B1', data[1])
worksheet.write_column('C1', data[2])

# Configure the chart. In simplest case we add one or more data series.
chart.add_series({'values': '=Sheet1!$A$1:$A$5'})
chart.add_series({'values': '=Sheet1!$B$1:$B$5'})
chart.add_series({'values': '=Sheet1!$C$1:$C$5'})

# Insert the chart into the worksheet.
```

```
worksheet.insert_chart('A7', chart)

workbook.close()
```



支持的图表类型是：

- `area`：创建区域（实线）样式图表。
- `bar`：创建条形样式（转置直方图）图表。
- `column`：创建列样式（直方图）图表。
- `line`：创建线型图表。
- `pie`：创建饼图样式图表。
- `doughnut`：创建圆环样式图表。
- `scatter`：创建散点图样式图表。
- `stock`：创建库存样式图表。
- `radar`：创建雷达样式图表。

某些图表类型也支持图表子类型：

```
workbook.add_chart({'type': 'bar', 'subtype': 'stacked'})
```

可用的子类型是：

```
area
    stacked
    percent_stacked

bar
    stacked
    percent_stacked
```

```
column
    stacked
    percent_stacked

scatter
    straight_with_markers
    straight
    smooth_with_markers
    smooth

radar
    with_markers
    filled
```

下面记录了所有图表类型共有的方法。有关图表的具体信息，请参阅 使用图表。

chart.add_series（）

- `add_series`（*选项*）

  将数据系列添加到图表。参数：**options**（*dict*） - 图表系列选项的字典。

在Excel中，图表**系列**是一组信息，用于定义绘制哪些数据，例如值，轴标签和格式。

对于XlsxWriter图表对象，该`add_series()`方法用于设置系列的属性：

```
chart.add_series({
    'categories': '=Sheet1!$A$1:$A$5',
    'values':     '=Sheet1!$B$1:$B$5',
    'line':       {'color': 'red'},
})

# Or using a list of values instead of category/value formulas:
#     [sheetname, first_row, first_col, last_row, last_col]
chart.add_series({
    'categories': ['Sheet1', 0, 0, 4, 0],
    'values':     ['Sheet1', 0, 1, 4, 1],
    'line':       {'color': 'red'},
})
```

如上所示，`categories`并且`values`可以采用范围公式，例如，`=Sheet1!$A$2:$A$7`或者在以编程方式生成范围时更有用的是具有零索引行/列值的列表。

可以设置的系列选项是：

- `values`：这是系列中最重要的属性，是每个图表对象的唯一必需选项。此选项将图表与其显示的工作表数据相链接。可以使用上面第一个示例中所示的公式或使用第二个示例中所示的值列表来设置数据范围。
- `categories`：这将设置图表类别标签。该类别与X轴大致相同。在大多数图表类型中，该`categories`属性是可选的，图表将仅假设一个顺序系列`1..n`。
- `name`：设置系列的名称。名称显示在公式栏中。对于非饼图/圆环图，它也会显示在图例中。name属性是可选的，如果未提供，则默认为 。名称也可以是公式，例如，包含工作表名称，行和列的列表。`Series 1..n``=Sheet1!$A$1``['Sheet1', 0, 0]`

- `line`：设置系列线型的属性，如颜色和宽度。请参见图表格式：行。
- `border`：设置系列的边框属性，如颜色和样式。请参见图表格式：边框。
- `fill`：设置系列的实心填充属性，例如颜色。请参见 图表格式：实心填充。
- `pattern`：设置系列的图案填充属性。请参见 图表格式：图案填充。
- `gradient`：设置系列的渐变填充属性。请参见 图表格式：渐变填充。
- `marker`：设置系列标记的属性，如样式和颜色。请参见图表系列选项：标记。
- `trendline`：设置系列趋势线的属性，如线性，多项式和移动平均类型。请参见 图表系列选项：趋势线。
- `smooth`：设置线系列的平滑属性。
- `y_error_bars`：设置图表系列的垂直误差范围。请参见 图表系列选项：错误栏。
- `x_error_bars`：设置图表系列的水平误差范围。请参见 图表系列选项：错误栏。
- `data_labels`：设置系列的数据标签。请参见 图表系列选项：数据标签。
- `points`：设置系列中各个点的属性。请参见 图表系列选项：点。
- `invert_if_negative`：反转负值的填充颜色。通常仅适用于柱形图和条形图。
- `overlap`：在条形图/柱形图中设置系列之间的重叠。范围是+/- 100.默认值为0：

  ```
  chart.add_series({
      'categories': '=Sheet1!$A$1:$A$5',
      'values':     '=Sheet1!$B$1:$B$5',
      'overlap':    10,
  })
  ```

  请注意，只需将`overlap`属性应用于图表中的一个系列。

- `gap`：在条形图/柱形图中设置系列之间的间隙。范围是0到500.默认值是150：

  ```
  chart.add_series({
      'categories': '=Sheet1!$A$1:$A$5',
      'values':     '=Sheet1!$B$1:$B$5',
      'gap':        200,
  })
  ```

  请注意，只需将`gap`属性应用于图表中的一个系列。

可以在图表中添加多个系列。实际上，有些图表类型`stock`需要它。Excel图表中的系列编号和顺序将与在XlsxWriter中添加它们的顺序相同。

也可以指定不连续的范围：

```
chart.add_series({
    'categories': '=(Sheet1!$A$1:$A$9,Sheet1!$A$14:$A$25)',
    'values':     '=(Sheet1!$B$1:$B$9,Sheet1!$B$14:$B$25)',
})
```

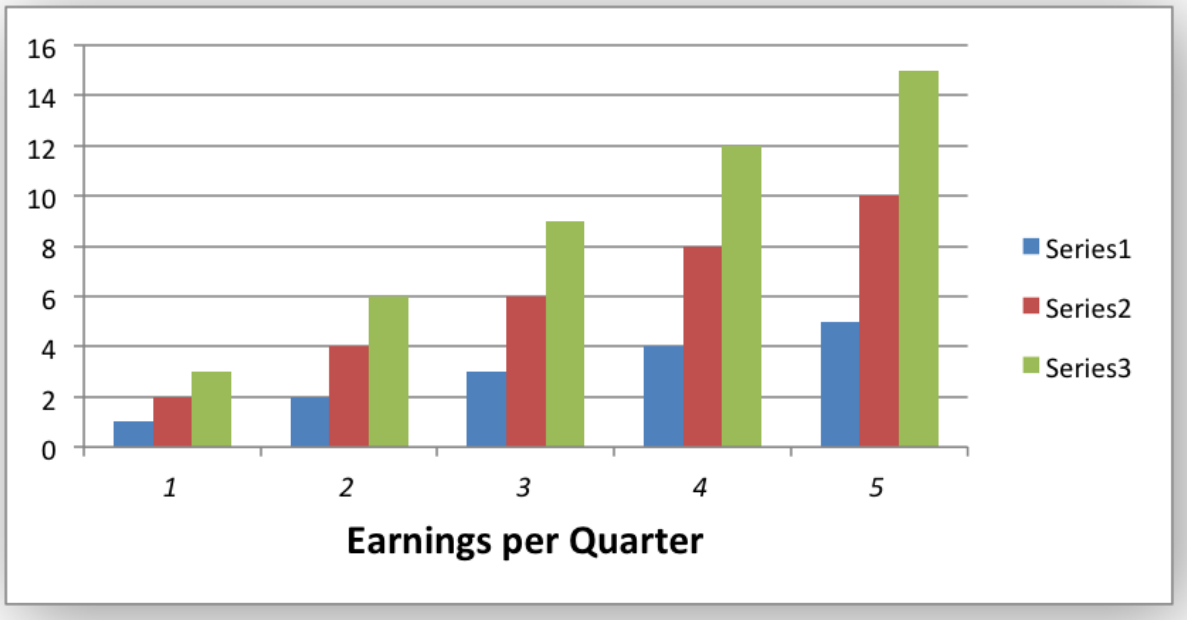
chart.set_x_axis（）

- `set_x_axis` (*选项*)

  设置图表X轴选项。参数：**options**（*dict*） - 轴选项字典。

该`set_x_axis()`方法用于设置X轴的属性：

```
chart.set_x_axis({
    'name': 'Earnings per Quarter',
    'name_font': {'size': 14, 'bold': True},
    'num_font':  {'italic': True },
})
```



可以设置的选项是：

```
name
name_font
name_layout
num_font
num_format
line
fill
pattern
gradient
min
max
minor_unit
major_unit
interval_unit
interval_tick
crossing
position_axis
```

```
reverse
log_base
label_position
label_align
major_gridlines
minor_gridlines
visible
date_axis
text_axis
minor_unit_type
major_unit_type
minor_tick_mark
major_tick_mark
display_units
display_units_visible
```

这些选项解释如下。某些属性仅适用于 **值**，**类别**或**日期**轴（在每种情况下都会注明）。有关Excel轴类型之间区别的说明，请参见 图表值和类别轴。

- `name`：设置轴的名称（也称为标题或标题）。名称显示在X轴下方。（适用于类别，日期和数值轴。）：

  ```
  chart.set_x_axis({'name': 'Earnings per Quarter'})
  ```

  此属性是可选的。默认设置是没有轴名称。

  名称也可以是公式，例如，`=Sheet1!$A$1`包含工作表名称，行和列的列表。`['Sheet1', 0, 0]`

- `name_font`：设置轴名称的字体属性。（适用于类别，日期和数值轴。）：

  ```
  chart.set_x_axis({'name_font': {'bold': True, 'italic': True}})
  ```

  有关字体属性的更多详细信息，请参见图表字体部分。

- `name_layout`：以图表相对单位设置轴标题的位置。（适用于类别，日期和数值轴。）：`(x, y)`

  ```
  chart.set_x_axis({
      'name': 'X axis',
      'name_layout': {
          'x': 0.34,
          'y': 0.85,
      }
  })
  ```

  有关详细信息，请参阅图表布局部分。

- `num_font`：设置轴编号的字体属性。（适用于类别，日期和数值轴。）：

  ```
  chart.set_x_axis({'name_font': {'bold': True, 'italic': True}})
  ```

  有关字体属性的更多详细信息，请参见图表字体部分。

- `num_format`：设置轴的数字格式。（适用于类别，日期和数值轴。）：

```
chart.set_x_axis({'num_format': '#,##0.00'})
chart.set_y_axis({'num_format': '0.00%'})
```

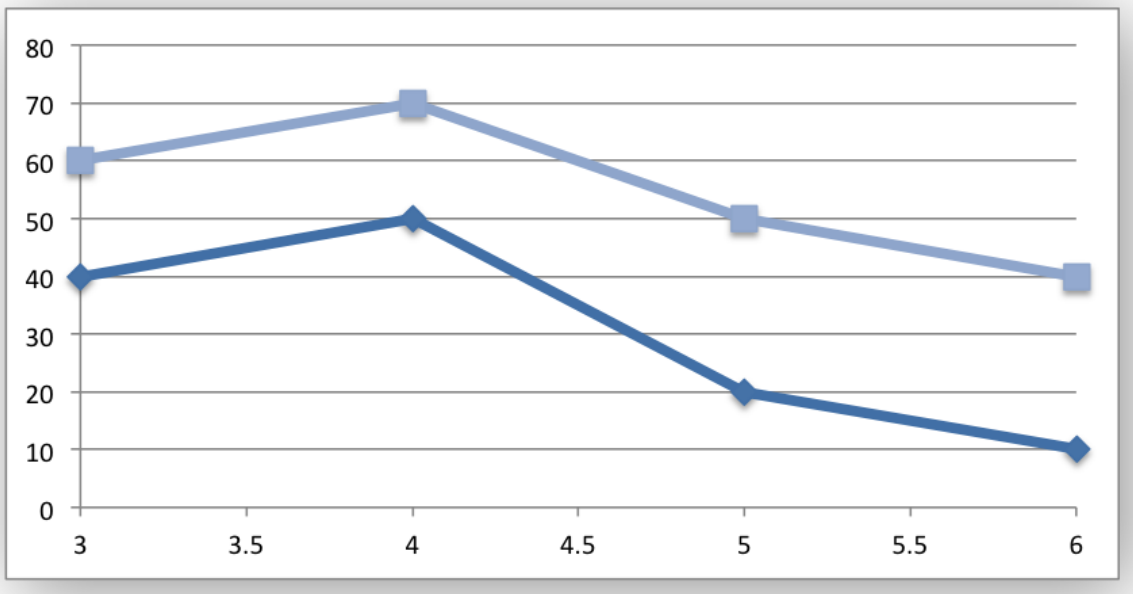
数字格式类似于工作表单元格格式 `num_format` ，但不能使用格式索引。必须使用显式格式字符串，如上所示。有关 `set_num_format()` 更多信息，请参阅

- `line`：设置轴线类型的属性，例如颜色和宽度。请参见图表格式：行：

```
chart.set_x_axis({'line': {'none': True}})
```

- `fill`：设置轴的实心填充属性，例如颜色。请参见 图表格式：实心填充。请注意，在Excel中，轴填充应用于轴编号的区域，而不是轴边界框的区域。该背景是从图表填充中设置的。
- `pattern`：设置轴的图案填充属性。请参见 图表格式：图案填充。
- `gradient`：设置轴的渐变填充属性。请参见 图表格式：渐变填充。
- `min`：设置轴范围的最小值。（仅适用于数值和日期轴。）：

```
chart.set_x_axis({'min': 3, 'max': 6})
```



- `max`：设置轴范围的最大值。（仅适用于数值和日期轴。）
- `minor_unit`：设置轴单位范围内次要单位的增量。（仅适用于数值和日期轴。）：

```
chart.set_x_axis({'minor_unit': 0.4, 'major_unit': 2})
```

- `major_unit`：设置轴范围内主要单位的增量。（仅适用于数值和日期轴。）
- `interval_unit`：设置类别轴的间隔单位。应该是一个整数值。（仅适用于类别轴。）：

```
chart.set_x_axis({'interval_unit': 5})
```

- `interval_tick`：设置类别轴的滴答间隔。应该是一个整数值。（仅适用于类别轴。）：

```
chart.set_x_axis({'interval_tick': 2})
```

- `crossing`：设置y轴穿过x轴的位置。（适用于所有轴。）

  该`crossing`值可以是`'max'`在最大轴值处设置交叉的字符串，也可以是数值：

```
chart.set_x_axis({'crossing': 3})
chart.set_y_axis({'crossing': 'max'})
```

  对于类别轴，数值必须是整数，以表示轴穿过的类别编号。对于值和日期轴，它可以具有与轴关联的任何值。另请参见图表值和类别轴。

  如果省略交叉（默认），则交叉将由Excel根据图表数据自动设置。

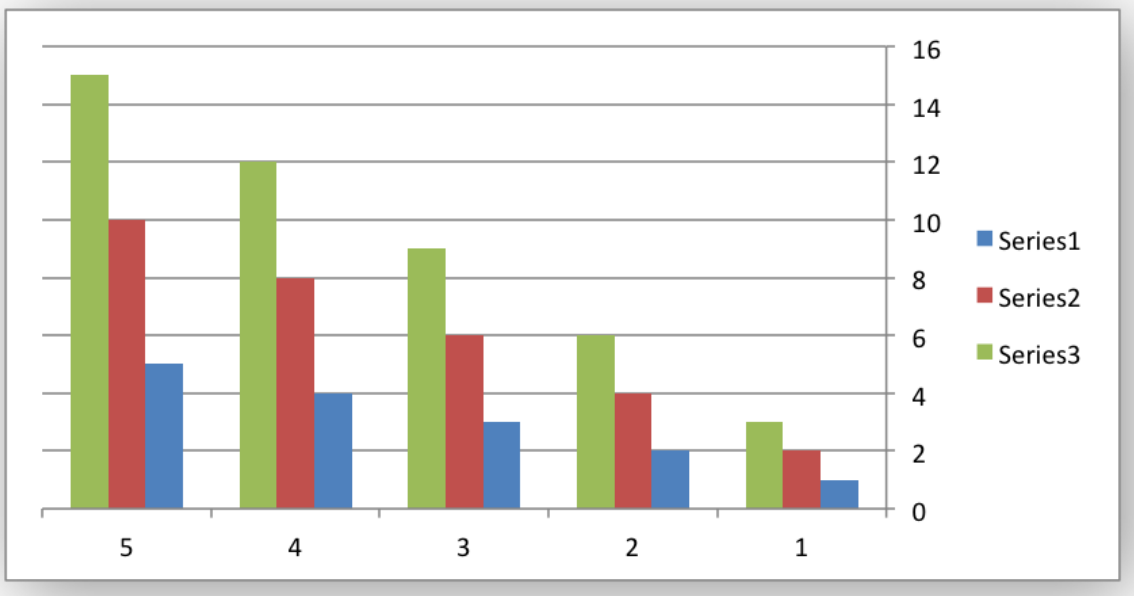
- `position_axis`：将轴定位在轴刻度标记上或之间。（仅适用于类别轴。）

  有两个允许的值`on_tick`和`between`：

```
chart.set_x_axis({'position_axis': 'on_tick'})
chart.set_x_axis({'position_axis': 'between'})
```

- `reverse`：反转轴类别或值的顺序。（适用于类别，日期和数值轴。）：

```
chart.set_x_axis({'reverse': True})
```



- `log_base`：设置轴范围的日志基数。（仅适用于数值轴。）：

```
chart.set_y_axis({'log_base': 10})
```

- `label_position`：设置轴的“轴标签”位置。可以使用以下职位：

  ```
  next_to (the default)
  high
  low
  none
  ```

  例如：

  ```
  chart.set_x_axis({'label_position': 'high'})
  chart.set_y_axis({'label_position': 'low'})
  ```

- `label_align`：将“轴标签”与轴对齐。（仅适用于类别轴。）

  可以使用以下Excel对齐：

  ```
  center (the default)
  right
  left
  ```

  例如：

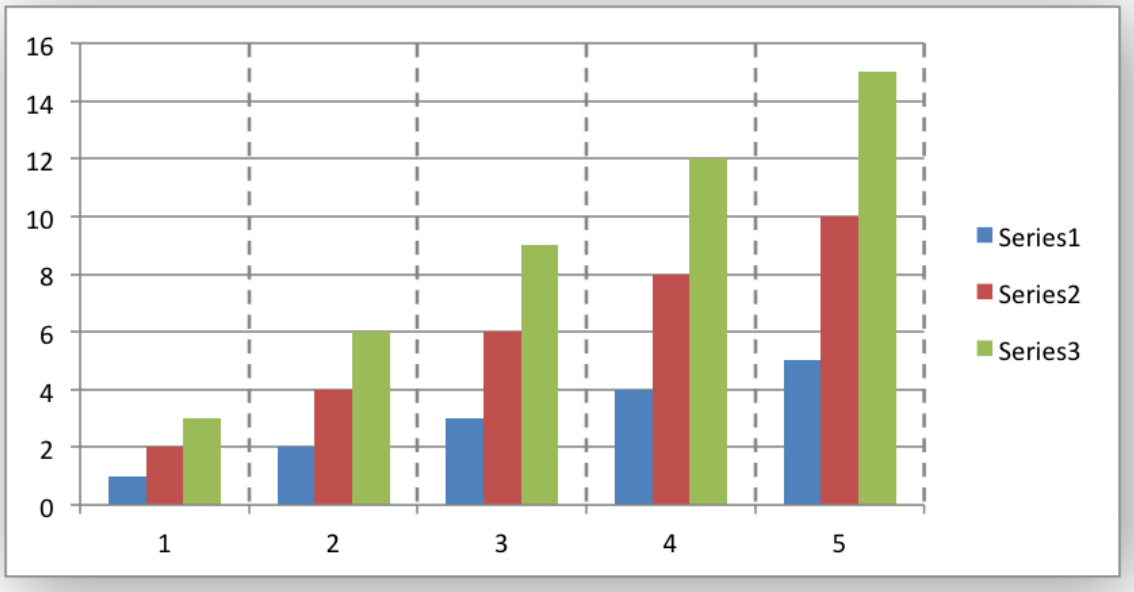
  ```
  chart.set_x_axis({'label_align': 'left'})
  ```

- `major_gridlines`：配置轴的主要网格线。可用的属性是：

  ```
  visible
  line
  ```

  例如：

  ```
  chart.set_x_axis({
      'major_gridlines': {
          'visible': True,
          'line': {'width': 1.25, 'dash_type': 'dash'}
      },
  })
  ```



该`visible`属性通常用于X轴，但它取决于图表的类型。

该`line`属性设置网格线属性，如颜色和宽度。请参见图表格式。

- `minor_gridlines`：这与`major_gridlines`上面的选项相同。

  `visible`默认情况下，对于所有图表类型，次要网格线属性均已关闭。

- `visible`：配置轴的可见性：

```
chart.set_y_axis({'visible': False})
```

  默认情况下，轴是可见的。

- `date_axis`：此选项用于将具有日期或时间数据的类别轴视为日期轴。（仅适用于日期类别轴。）：

```
chart.set_x_axis({'date_axis': True})
```

  此选项还允许您设置`max`和`min`用于没有被Excel允许非日期类别轴类别轴值。

  有关更多详细信息，请参见日期类别轴。

- `text_axis`：此选项用于将类别轴明确视为文本轴。（仅适用于类别轴。）：

```
chart.set_x_axis({'text_axis': True})
```

- `minor_unit_type`：对于`date_axis`轴，请参见上文，此选项用于设置次要单位的类型。（仅适用于日期类别轴。）：

```
chart.set_x_axis({
    'date_axis': True,
    'minor_unit': 4,
    'minor_unit_type': 'months',
})
```

- `major_unit_type`：与`minor_unit_type`上面相同，但主轴单位类型。
- `minor_tick_mark`：将轴次要刻度标记类型/位置设置为以下值之一：

  ```
  none
  inside
  outside
  cross   (inside and outside)
  ```

  例如：

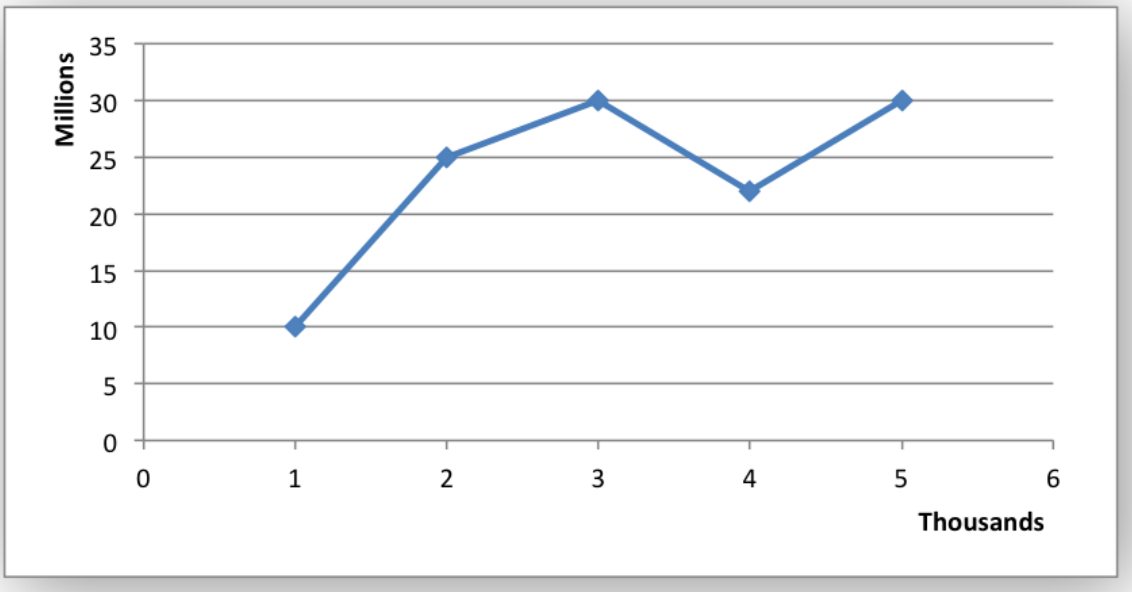
  ```
  chart.set_x_axis({'major_tick_mark': 'none',
                    'minor_tick_mark': 'inside'})
  ```

- `major_tick_mark`：相同`minor_tick_mark`，见上文，但主轴蜱。
- `display_units`：设置轴的显示单位。如果轴数非常大但您不想用科学计数法表示它们，这可能很有用。可用的显示单位是：

  ```
  hundreds
  thousands
  ten_thousands
  hundred_thousands
  millions
  ten_millions
  hundred_millions
  billions
  trillions
  ```

  仅适用于数值轴：

  ```
  chart.set_x_axis({'display_units': 'thousands'})
  chart.set_y_axis({'display_units': 'millions'})
  ```



- `display_units_visible`：控制前一个选项打开的人机界面的可见性。默认情况下，此选项处于启用状 （仅适用于数值轴。）：

```
chart.set_x_axis({'display_units': 'hundreds',
                  'display_units_visible': False})
```

chart.set_y_axis（）

- `set_y_axis` （*选项*）

  设置图表Y轴选项。参数：**options**（*dict*） - 轴选项字典。

该`set_y_axis()`方法用于设置Y轴的属性。

可以设置的属性与`set_x_axis`上面相同，见上文。

chart.set_x2_axis（）

- `set_x2_axis` （*选项*）

  设置图表辅助X轴选项。参数：**options**（*dict*） - 轴选项字典。

该`set_x2_axis()`方法用于设置辅助X轴的属性，请参阅`chart_secondary_axes()`。

可以设置的属性与`set_x_axis`上面相同，见上文。

该轴的默认属性是：

```
'label_position': 'none',
'crossing':       'max',
'visible':        False,
```

chart.set_y2_axis（）

- `set_y2_axis`（*选项*）

  设置图表辅助Y轴选项。参数：**options**（*dict*） - 轴选项字典。

该`set_y2_axis()`方法用于设置辅助Y轴的属性，请参阅`chart_secondary_axes()`。

可以设置的属性与`set_x_axis`上面相同，见上文。

该轴的默认属性是：

```
'major_gridlines': {'visible': True}
```

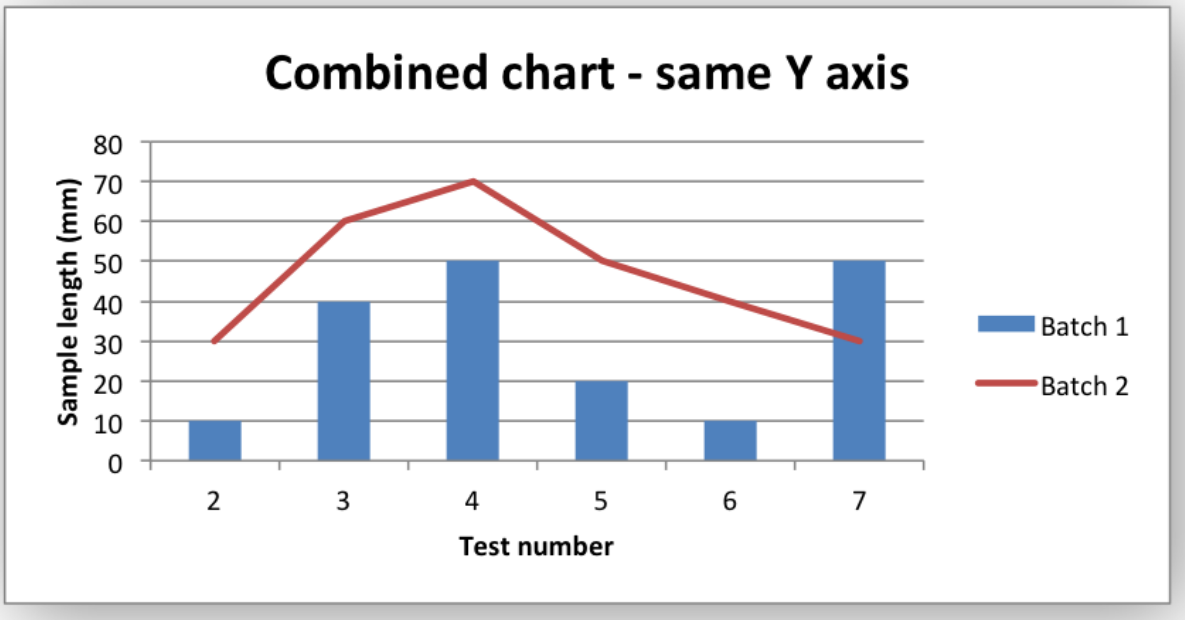
chart.combine（）

- `combine`（*图*）

  合并两个不同类型的图表。参数：**chart** - 使用创建的图表对象`add_chart()`。

图表`combine()`方法用于组合两个不同类型的图表，例如列和折线图：

```
# Create a primary chart.
column_chart = workbook.add_chart({'type': 'column'})
column_chart.add_series({...})

# Create a secondary chart.
line_chart = workbook.add_chart({'type': 'line'})
line_chart.add_series({...})

# Combine the charts.
column_chart.combine(line_chart)
```



有关详细信息，请参阅组合图表部分。

chart.set_size（）

该`set_size()`方法用于设置图表的尺寸。可以设置的大小属性是：

```
width
height
x_scale
y_scale
x_offset
y_offset
```

在`width`和`height`以像素为单位。默认图表宽度x高度为480 x 288像素。可以通过设置`width`和`height`或通过设置`x_scale`和来修改图表的大小`y_scale`：

```
chart.set_size({'width': 720, 'height': 576})
# Same as:
chart.set_size({'x_scale': 1.5, 'y_scale': 2})
```

的`x_offset`和`y_offset`，它被插入到细胞中的图表的左上角的位置。

注意：`x_offset`和`y_offset`参数也可以通过以下`insert_chart()`方法设置：

```
worksheet.insert_chart('E2', chart, {'x_offset': 25, 'y_offset': 10})
```

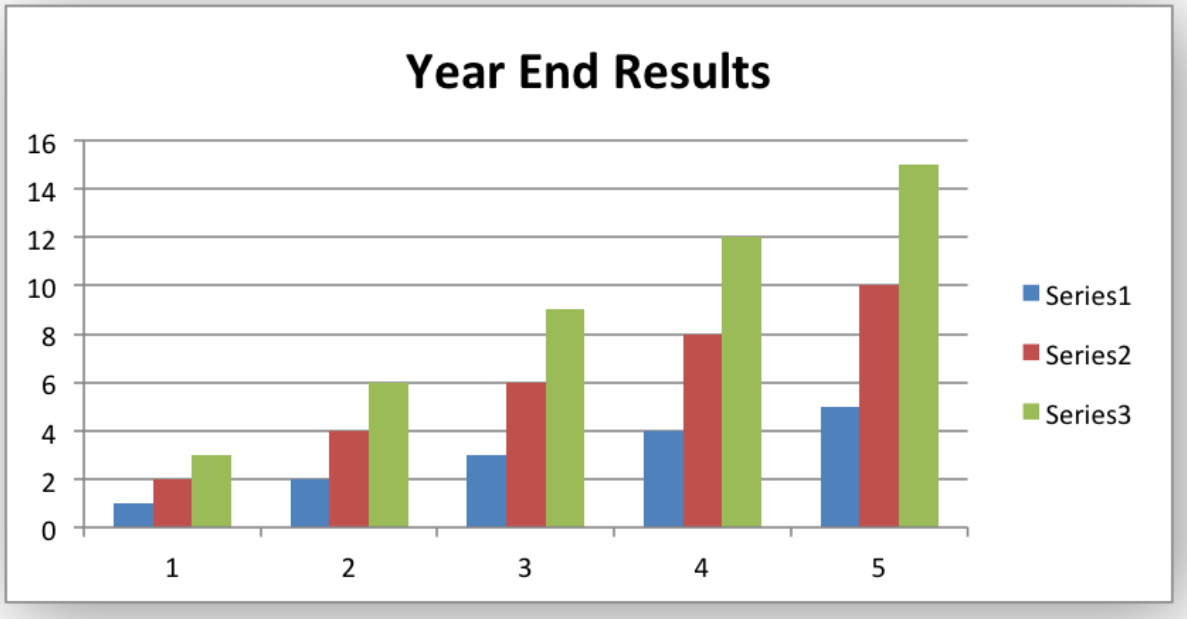
chart.set_title（）

- `set_title` (*选项*)

  设置图表标题选项。参数：**options**（*dict*） - 图表大小选项的字典。

该`set_title()`方法用于设置图表标题的属性：

```
chart.set_title({'name': 'Year End Results'})
```



可以设置的属性是：

- `name`：设置图表的名称（标题）。名称显示在图表上方。名称也可以是公式，例如，`=Sheet1!$A$1`包含工作表名称，行和列的列表。name属性是可选的。默认是没有图表标题。`['Sheet1', 0, 0]`
- `name_font`：设置图表标题的字体属性。请参见 图表字体。
- `overlay`：允许标题覆盖在图表上。通常与下面的布局属性一起使用。
- `layout`：以图表相对单位设置标题的位置：`(x, y)`

  ```
  chart.set_title({
      'name': 'Title',
      'overlay': True,
      'layout': {
          'x': 0.42,
          'y': 0.14,
      }
  })
  ```

  有关详细信息，请参阅图表布局部分。

- `none`：默认情况下，Excel会为具有单个系列和用户定义的系列名称的图表添加自动图表标题。该`none`选项会关闭此默认标题。它还会关闭所有其他`set_title()`选项：

```
chart.set_title({'none': True})
```

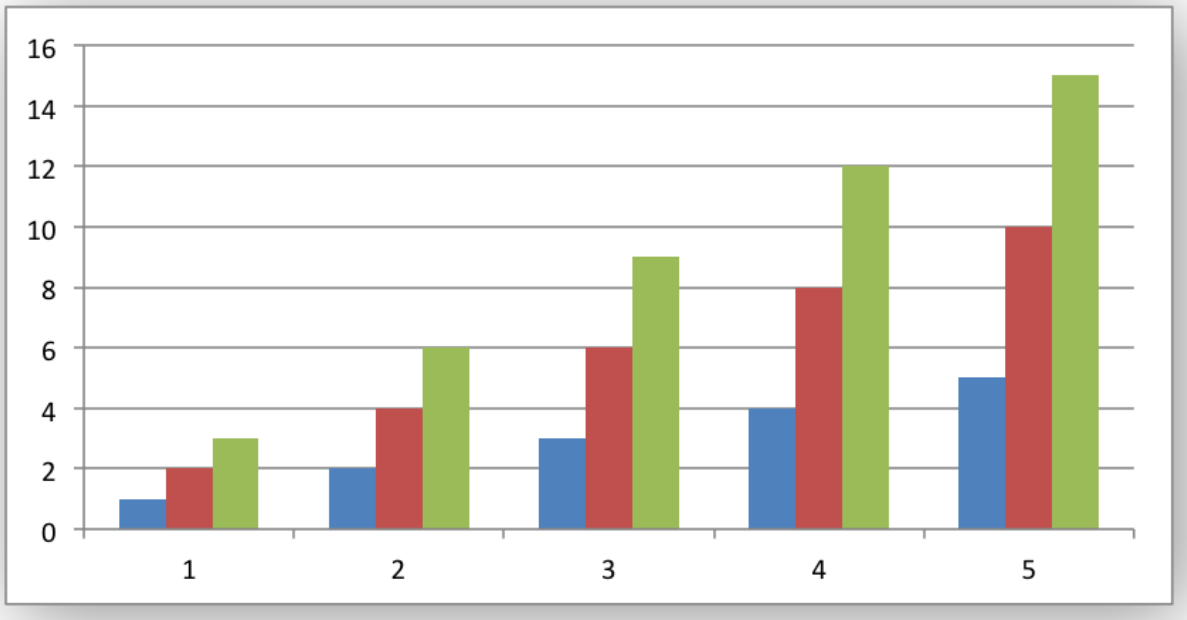
chart.set_legend（）

- `set_legend`（*选项*）

  设置图表图例选项。参数：**options**（*dict*） - 图表图例选项的字典。

该`set_legend()`方法用于设置图表图例的属性。例如，它可用于关闭默认图表图例：

```
chart.set_legend({'none': True})
```



可以设置的选项是：

```
none
position
font
border
fill
pattern
gradient
delete_series
layout
```

- `none`：在Excel图表中，默认情况下图例处于启用状态。该`none`选项会关闭图表图例：

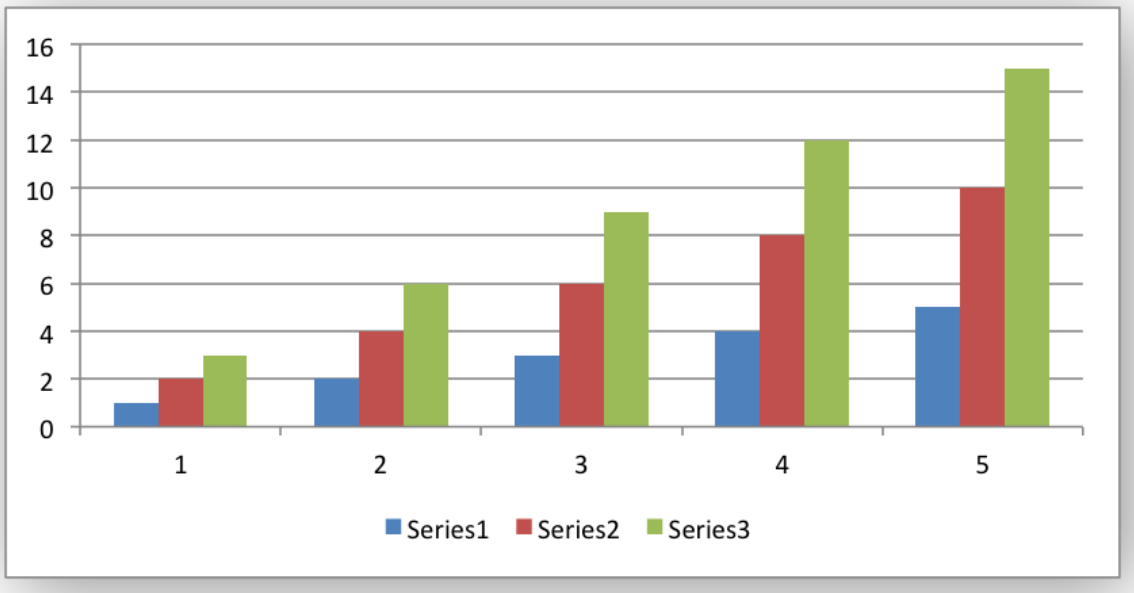
  ```
  chart.set_legend({'none': True})
  ```

  为了向后兼容，还可以通过`position`属性关闭图例：

```
chart.set_legend({'position': 'none'})
```

- `position`：设置图表图例的位置：

```
chart.set_legend({'position': 'bottom'})
```



默认图例位置是`right`。可用的职位是：

```
top
bottom
left
right
overlay_left
overlay_right
none
```

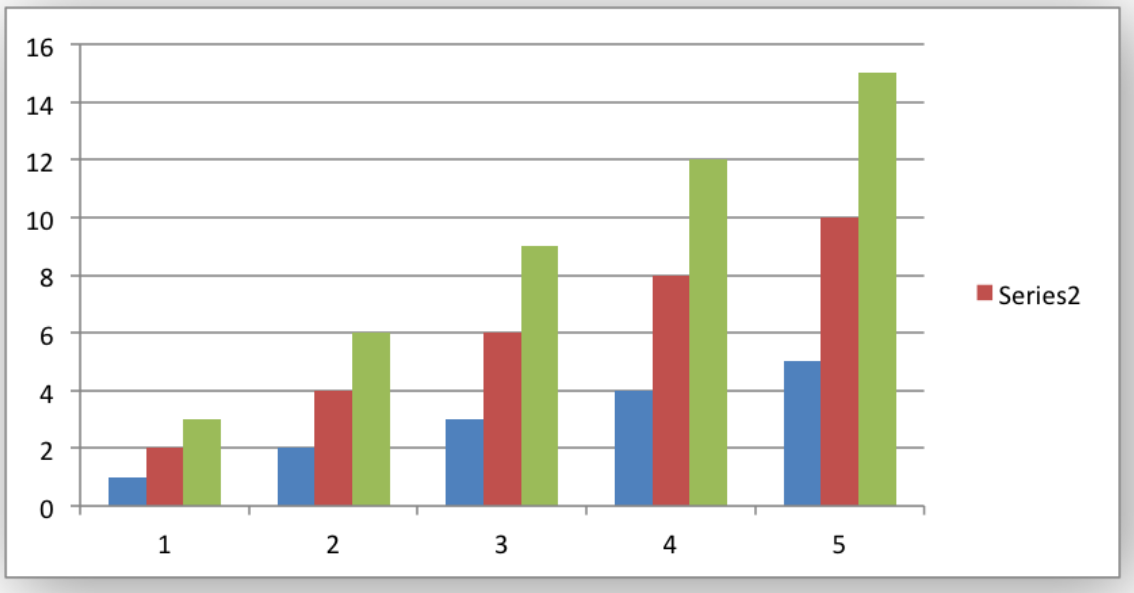
- `font`：设置图表图例的字体属性：

```
chart.set_legend({'font': {'size': 9, 'bold': True}})
```

有关字体属性的更多详细信息，请参见图表字体部分。

- `border`：设置图例的边框属性，例如颜色和样式。请参见图表格式：边框。
- `fill`：设置图例的实心填充属性，例如颜色。请参见 图表格式：实心填充。
- `pattern`：设置图例的图案填充属性。请参见 图表格式：图案填充。
- `gradient`：设置图例的渐变填充属性。请参见 图表格式：渐变填充。
- `delete_series`：这允许您从图例中删除一个或多个系列（系列仍将显示在图表上）。此属性将列表作为参数，系列为零索引：

```
# Delete/hide series index 0 and 2 from the legend.
chart.set_legend({'delete_series': [0, 2]})
```



- `layout`：以图表相对单位设置图例的位置：`(x, y)`

```
chart.set_legend({
    'layout': {
        'x':      0.80,
        'y':      0.37,
        'width':  0.12,
        'height': 0.25,
    }
})
```

有关详细信息，请参阅图表布局部分。

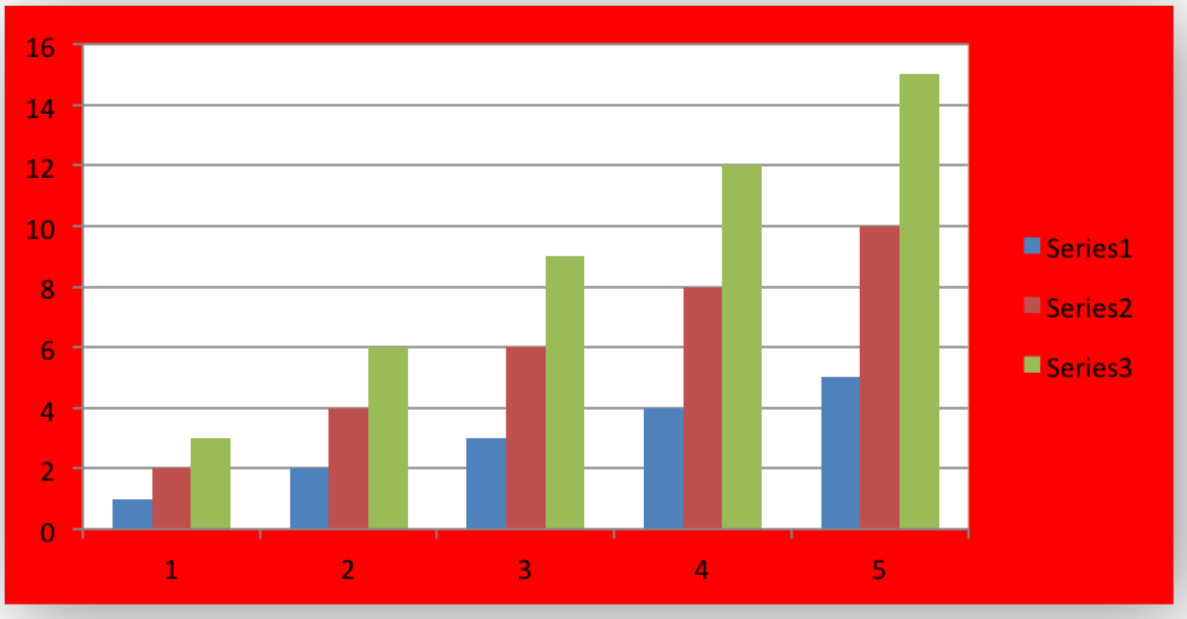
chart.set_chartarea（）

- `set_chartarea` (*选项*)

  设置图表区域选项。参数：**options**（*dict*） - 图表区域选项的字典。

该`set_chartarea()`方法用于设置图表区域的属性。在Excel中，图表区域是图表背后的背景区域：

```
chart.set_chartarea({
    'border': {'none': True},
    'fill':   {'color': 'red'}
})
```



可以设置的属性是：

- `border`：设置图表区域的边框属性，例如颜色和样式。请参见图表格式：边框。
- `fill`：设置图表区域的实心填充属性，例如颜色。请参见 图表格式：实心填充。
- `pattern`：设置图表区域的图案填充属性。请参见 图表格式：图案填充。
- `gradient`：设置图表区域的渐变填充属性。请参见 图表格式：渐变填充。

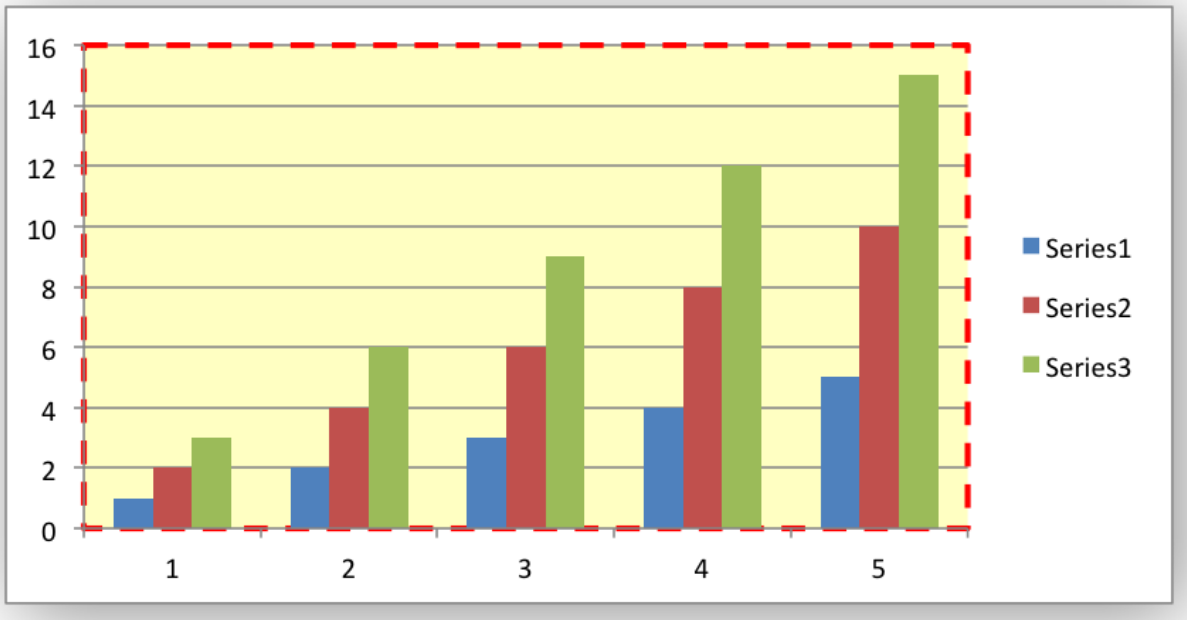
chart.set_plotarea（）

- `set_plotarea`（*选项*）

  设置绘图区域选项。参数：**options**（*dict*） - 绘图区域选项的字典。

该`set_plotarea()`方法用于设置图表的绘图区域的属性。在Excel中，绘图区域是绘制图表系列的轴之间的区域：

```
chart.set_plotarea({
    'border': {'color': 'red', 'width': 2, 'dash_type': 'dash'},
    'fill':   {'color': '#FFFFC2'}
})
```



可以设置的属性是：

- `border`：设置plotarea的边框属性，如颜色和样式。请参见图表格式：边框。
- `fill`：设置plotarea的实心填充属性，例如颜色。请参见 图表格式：实心填充。
- `pattern`：设置plotarea的图案填充属性。请参见 图表格式：图案填充。
- `gradient`：设置plotarea的渐变填充属性。请参见 图表格式：渐变填充。
- `layout`：以图表相对单位设置plotarea 的位置：`(x, y)`

```
chart.set_plotarea({
    'layout': {
        'x':      0.13,
        'y':      0.26,
        'width':  0.73,
        'height': 0.57,
    }
})
```

有关详细信息，请参阅图表布局部分。

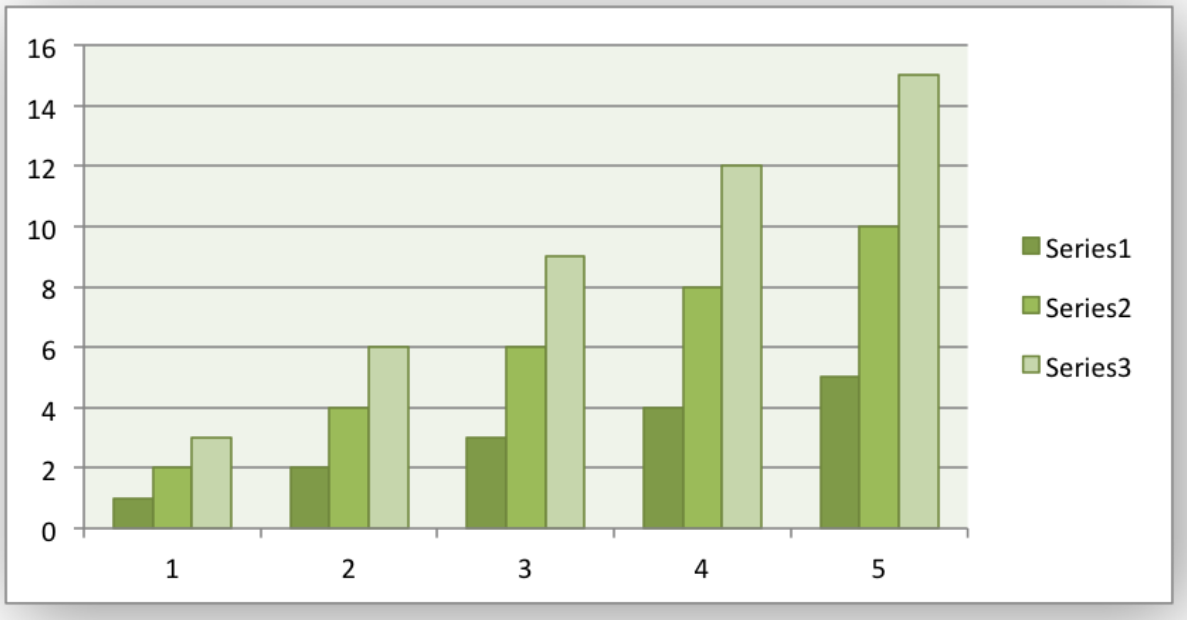
chart.set_style（）

- `set_style` (*style_id*)

  设置图表样式类型。参数：**style_id**（*int*） - 表示图表样式的索引。

该`set_style()`方法用于将图表的样式设置为Excel中“设计”选项卡上可用的48种内置样式之一：

```
chart.set_style(37)
```



样式索引号从左上角的1开始计算。默认样式为2。

注意

在Excel 2013中，Excel中“设计”选项卡的“样式”部分显示了以前版本的Excel中所谓的“布局”。这些布局未以文件格式定义。它们是基本图表类型的修改集合。可以使用XlsxWriter Chart API复制它们，但不能通过该 `set_style()` 方法定义它们。

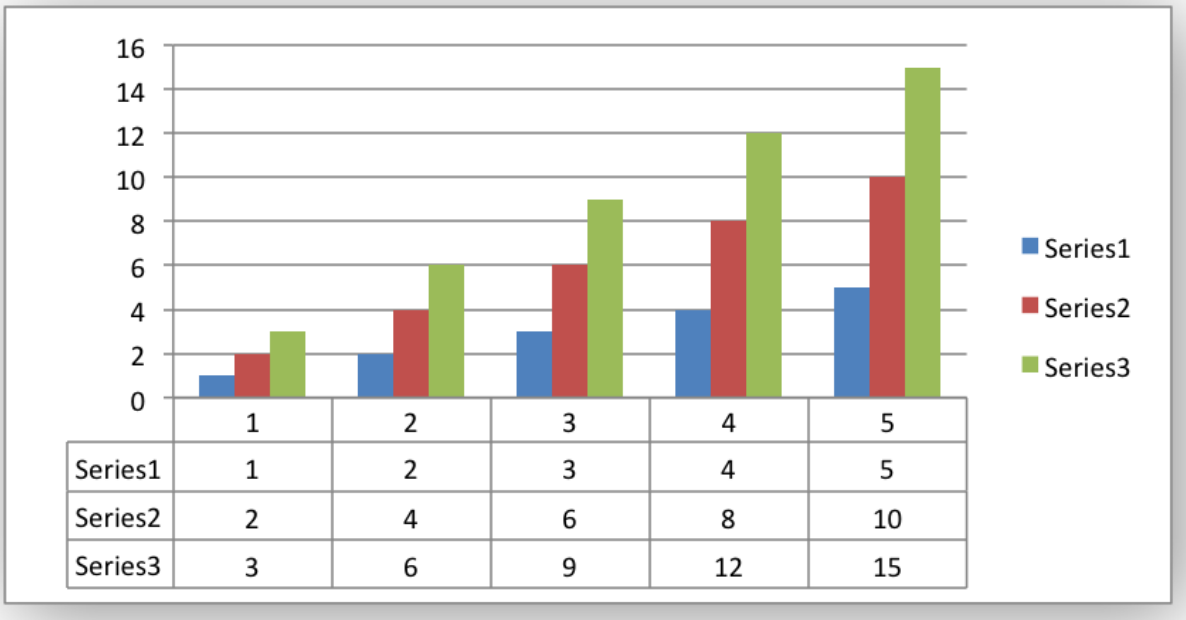
chart.set_table（）

- `set_table` （*选项*）

  设置轴数据表的属性。参数：**options**（*dict*） - 轴表选项的字典。

该 `set_table()` 方法在水平轴下方添加一个数据表，其中包含用于绘制图表的数据：

```
chart.set_table()
```



	1	2	3	4	5
Series1	1	2	3	4	5
Series2	2	4	6	8	10
Series3	3	6	9	12	15

可用选项的默认值为：

```
'horizontal': True   # Display vertical lines in the table.
'vertical':   True   # Display horizontal lines in the table.
'outline':    True   # Display an outline in the table.
'show_keys':  False  # Show the legend keys with the table data.
'font':       {}     # Standard chart font properties.
```

例如：

```
chart.set_table({'show_keys': True})
```

数据表只能显示条形图，柱形图，折线图，面积图和股票图。有关字体属性的更多详细信息，请参见图表字体部分。

chart.set_up_down_bars（）

- `set_up_down_bars`（*选项*）

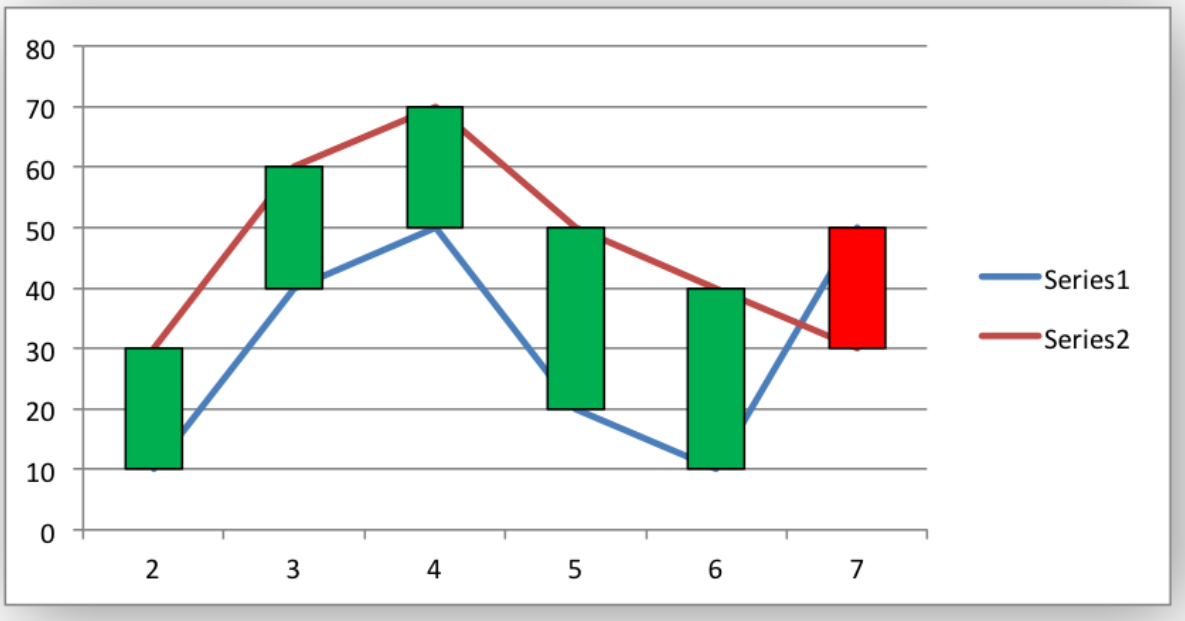
  设置图表上下栏的属性。参数：**options**（*dict*） - 选项词典。

该`set_up_down_bars()`方法将“上下”栏添加到折线图以指示第一个和最后一个数据系列之间的差异：

```
chart.set_up_down_bars()
```

它可以格式化上下酒吧添加`fill`，`pattern`或`gradient`和`border`性能如果需要的话。请参见 图表格式：

```
chart.set_up_down_bars({
    'up': {
        'fill':   {'color': '#00B050'},
        'border': {'color': 'black'}
    },
    'down': {
        'fill':   {'color': 'red'},
        'border': {'color': 'black'},
    },
})
```



上下栏只能应用于折线图和股票图表（默认情况下）。

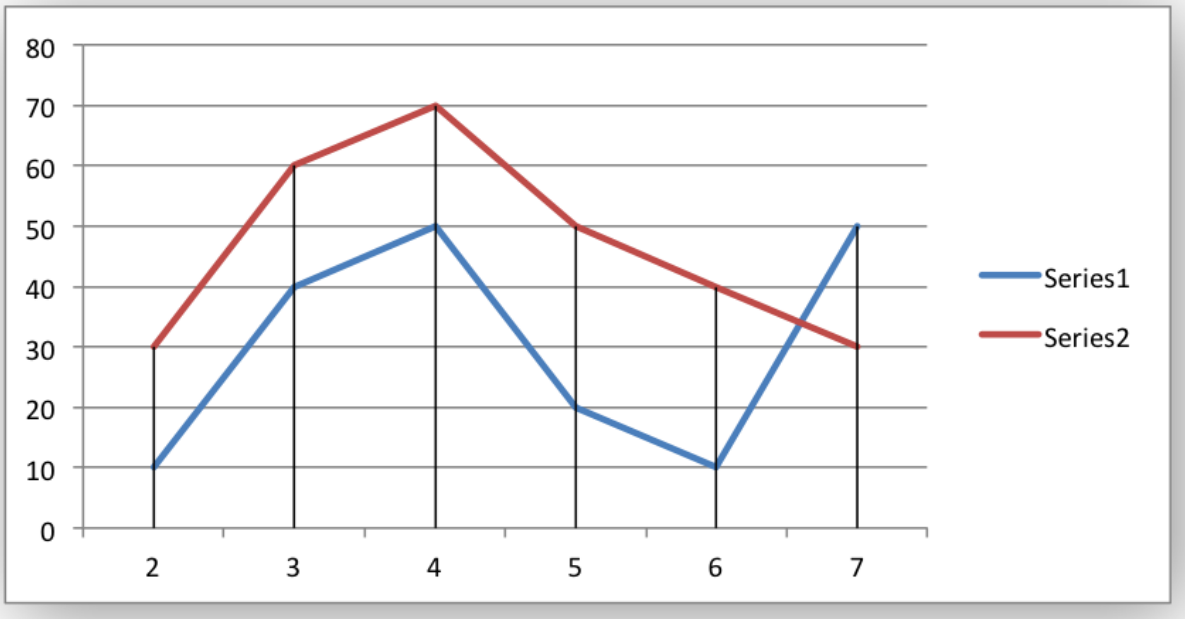
chart.set_drop_lines（）

- `set_drop_lines`（*选项*）

  设置图表下拉线的属性。参数：**options**（*dict*） - 选项词典。

该`set_drop_lines()`方法将Drop Lines添加到图表中以显示数据中点的Category值：

```
chart.set_drop_lines()
```



`line` 如果需要，可以格式化Drop Line 属性。请参见 图表格式：

```
chart.set_drop_lines({'line': {'color': 'red',
                               'dash_type': 'square_dot'}})
```

掉线仅适用于直线，面积和股票图表。

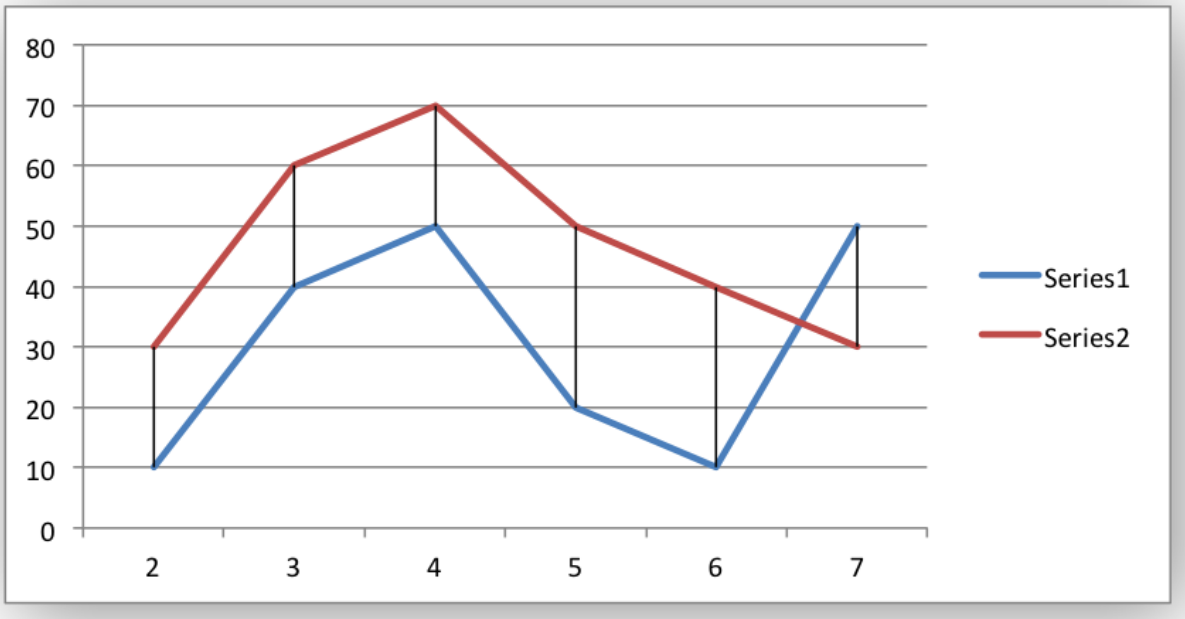
chart.set_high_low_lines（）

- `set_high_low_lines` （*选项*）

  设置图表高低线的属性。参数：**options**（*dict*） - 选项词典。

该 `set_high_low_lines()` 方法将高 - 低线添加到图表，以显示类别中点的最大值和最小值：

```
chart.set_high_low_lines()
```



`line` 如果需要，可以格式化High-Low Line 属性。请参见 图表格式:

```
chart.set_high_low_lines({
    'line': {
        'color': 'red',
        'dash_type': 'square_dot'
    }
})
```

高 - 低线仅在线和股票图表中可用。

chart.show_blanks_as（）

- `show_blanks_as` (*选项*)

  设置在图表中显示空白数据的选项。参数：**option**（*string*） - 表示显示选项的字符串。

该 `show_blanks_as()` 方法控制空白数据在图表中的显示方式:

```
chart.show_blanks_as('span')
```

可用选项包括:

```
'gap'   # Blank data is shown as a gap. The default.
'zero'  # Blank data is displayed as zero.
'span'  # Blank data is connected with a line.
```

chart.show_hidden_data（）

- `show_hidden_data()`

  在隐藏的行或列的图表上显示数据。

在图表上的隐藏行或列中显示数据：

```
chart.show_hidden_data()
```

chart.set_rotation（）

- `set_rotation`（*轮换*）

  设置饼图/圆环图旋转。参数：**rotation**（*int*） - 旋转角度。

该`set_rotation()`方法用于设置饼图/圆环图的第一段的旋转。这具有旋转整个图表的效果：

```
chart->set_rotation(90)
```

旋转角度必须在该范围内。`0 <= rotation <= 360`

此选项仅适用于饼图/圆环图。

chart.set_hole_size（）

- `set_hole_size`（*大小*）

  设置圆环图表孔尺寸。参数：**size**（*int*） - 孔的大小百分比。

该`set_hole_size()`方法用于设置圆环图的孔尺寸：

```
chart->set_hole_size(33)
```

孔尺寸的值必须在范围内。`10 <= size <= 90`

此选项仅适用于圆环图。

chartsheet

```
chartsheet = workbook.add_chartsheet()
chart      = workbook.add_chart({'type': 'bar'})

# Configure the chart.

chartsheet.set_chart(chart)
```

helpful function

xl_rowcol_to_cell（）

- `xl_rowcol_to_cell`（*row*，*col*[，*row_abs*，*col_abs*]）

将零索引行和列单元格引用转换为A1样式字符串。参数：**row**（*int*） - 单元格行。**col**（*int*） - 单元格列。**row_abs**（*bool*） - 使行绝对的可选标志。**col_abs**（*bool*） - 使列绝对的可选标志。返回类型：A1风格的字符串。

该`xl_rowcol_to_cell()`函数将零索引的行和列单元格值转换为`A1`样式字符串：

```
cell = xl_rowcol_to_cell(0, 0)   # A1
cell = xl_rowcol_to_cell(0, 1)   # B1
cell = xl_rowcol_to_cell(1, 0)   # A2
```

可选参数`row_abs`和`col_abs`可用于指示该行或列是绝对的：

```
str = xl_rowcol_to_cell(0, 0, col_abs=True)                # $A1
str = xl_rowcol_to_cell(0, 0, row_abs=True)                # A$1
str = xl_rowcol_to_cell(0, 0, row_abs=True, col_abs=True)  # $A$1
```

xl_cell_to_rowcol（）

- `xl_cell_to_rowcol`（*cell_str* ）

  将A1表示法中的单元格引用转换为零索引行和列。参数：**cell_str**（*string*） - A1样式字符串，绝对或相对。返回类型：（row，col）的整数元组

该`xl_cell_to_rowcol()`函数将`A1`符号中的Excel单元格引用转换为基于零的行和列。该函数还将处理Excel的绝对值`$`，单元格表示法：

```
(row, col) = xl_cell_to_rowcol('A1')    # (0, 0)
(row, col) = xl_cell_to_rowcol('B1')    # (0, 1)
(row, col) = xl_cell_to_rowcol('C2')    # (1, 2)
(row, col) = xl_cell_to_rowcol('$C2')   # (1, 2)
(row, col) = xl_cell_to_rowcol('C$2')   # (1, 2)
(row, col) = xl_cell_to_rowcol('$C$2')  # (1, 2)
```

xl_col_to_name（）

- `xl_col_to_name`（*col* [，*col_abs*]）

  将零索引列单元格引用转换为字符串。参数：**col**（*int*） - 单元格列。**col_abs**（*bool*） - 使列绝对的可选标志。返回类型：列样式字符串。

所述`xl_col_to_name()`转换是零基于列引用的字符串：

```
column = xl_col_to_name(0)    # A
column = xl_col_to_name(1)    # B
column = xl_col_to_name(702)  # AAA
```

可选参数`col_abs`可用于指示列是否为绝对列：

```
column = xl_col_to_name(0, False)  # A
column = xl_col_to_name(0, True)   # $A
column = xl_col_to_name(1, True)   # $B
```

xl_range（）

- `xl_range`（*first_row*，*first_col*，*last_row*，*last_col* ）

  将零索引行和列单元格引用转换为A1：B1范围字符串。参数：**first_row**（*int*） - 第一个单元格行。**first_col**（*int*） - 第一个单元格列。**last_row**（*int*） - 最后一个单元格行。**last_col**（*int*） - 最后一个单元格列。返回类型：A1：B1样式范围字符串。

该`xl_range()`函数将基于零的行和列单元格引用转换为`A1:B1`样式范围字符串：

```
cell_range = xl_range(0, 0, 9, 0)  # A1:A10
cell_range = xl_range(1, 2, 8, 2)  # C2:C9
cell_range = xl_range(0, 0, 3, 4)  # A1:E4
```

xl_range_abs（）

- `xl_range_abs`（*first_row*，*first_col*，*last_row*，*last_col* ）

  将零索引行和列单元格引用转换为$A$1：$B$1绝对范围字符串。参数：**first_row**（*int*） - 第一个单元格行。**first_col**（*int*） - 第一个单元格列。**last_row**（*int*） - 最后一个单元格行。**last_col**（*int*） - 最后一个单元格列。返回类型：$A$1：$B$1样式范围字符串。

该`xl_range_abs()`函数将基于零的行和列单元格引用转换为绝对`$A$1:$B$1`样式范围字符串：

```
cell_range = xl_range_abs(0, 0, 9, 0)  # $A$1:$A$10
cell_range = xl_range_abs(1, 2, 8, 2)  # $C$2:$C$9
cell_range = xl_range_abs(0, 0, 3, 4)  # $A$1:$E$4
```

worksheet(print setup)

```
worksheet.set_landscape () # 水平
worksheet.set_portrait() #纵向
worksheet.set_page_view() #页面视图
worksheet.set_paper(index)
worksheet.center_horizontally () #水平对中打印页面
worksheet.set_margins(left,right,top,bottom) #默认0.75

worksheet.set_header() #设置打印的页眉标题和选项
    >>> worksheet.set_header('&LCiao&CBello&RCielo')
    ----------------------------------------------------------------
   |                                                                |
   | Ciao                    Bello                          Cielo |
   |                                                                |
    >>> worksheet.set_header('&L&[Picture]&C&[Picture]&R&[Picture]',
                       {'image_left':   'red.jpg',
                        'image_center': 'blue.jpg',
```

```
                        'image_right':  'yellow.jpg'})

worksheet.set_footer() #同header
worksheet.repeat_rows() #设置要在每个打印页面顶部重复的行数
worksheet.repeat_columns() #将列设置为在每个打印页面的左侧重复
worksheet.hide_gridlines() # 设置选项以隐藏屏幕和打印页面上的网格线
    # 0.不要隐藏网格线。
    # 1.仅隐藏打印的网格线。(default)
    # 2.隐藏屏幕和打印的网格线。

worksheet.print_row_col_headers() # 打印行号和列号
worksheet.hide_row_col_headers()
worksheet.print_area() # 设置打印区域
worksheet.print_across() #设置打印页面的顺序
worksheet.fit_to_pages(width,height)
worksheet.set_start_page(2)
worksheet.set_print_scale(10) #设置打印页面的比例 10-400
worksheet.set_h_pagebreaks([20,30]) #水平分页符
worksheet.set_v_pagebreaks() #垂直分页符
```

.set_paper()

index	纸张格式	纸张尺寸
0	打印机默认	打印机默认
1	信件	8 1/2 x 11英寸
2	信小	8 1/2 x 11英寸
3	小报	11 x 17英寸
4	莱杰	17 x 11英寸
5	法律	8 1/2 x 14英寸
6	声明	5 1/2 x 8 1/2英寸
7	行政人员	7 1/4 x 10 1/2英寸
8	A3	297 x 420毫米
9	A4	210 x 297毫米
10	A4小	210 x 297毫米
11	A5	148 x 210毫米
12	B4	250 x 354毫米
13	B5	182 x 257毫米
14	开本	8 1/2 x 13英寸
15	房间	215 x 275毫米
16	—	10x14英寸
17	—	11x17英寸
18	注意	8 1/2 x 11英寸
19	信封9	3 7/8 x 8 7/8
20	信封10	4 1/8 x 9 1/2
21	信封11	4 1/2 x 10 3/8
22	信封12	4 3/4 x 11
23	信封14	5 x 11 1/2
24	C尺寸表	—
25	D尺寸表	—
26	E尺寸表	—
27	信封DL	110 x 220毫米

index	纸张格式	纸张尺寸
28	信封C3	324 x 458毫米
29	信封C4	229 x 324毫米
30	信封C5	162 x 229毫米
31	信封C6	114 x 162毫米
32	信封C65	114 x 229毫米
33	信封B4	250 x 353毫米
34	信封B5	176 x 250毫米
35	信封B6	176 x 125毫米
36	信封	110 x 230毫米
37	君主	3.875 x 7.5英寸
38	信封	3 5/8 x 6 1/2英寸
39	折叠式	14 7/8 x 11英寸
40	German Std Fanfold	8 1/2 x 12英寸
41	德国法律Fanfold	8 1/2 x 13英寸

.set_header()

Control	Category	Description
&L	Justification	Left
&C		Center
&R		Right
&P	Information	Page number
&N		Total number of pages
&D		Date
&T		Time
&F		File name
&A		Worksheet name
&Z		Workbook path
&fontsize	Font	Font size
&"font,style"		Font name and style
&U		Single underline
&E		Double underline
&S		Strikethrough
&X		Superscript
&Y		Subscript
&[Picture]	Images	Image placeholder
&G		Same as &[Picture]

chart detail

本节介绍如何使用图表类的一些选项和功能 。

本节中的大多数示例都基于以下程序的变体：

```
import xlsxwriter

workbook = xlsxwriter.Workbook('chart_line.xlsx')
worksheet = workbook.add_worksheet()

# Add the worksheet data to be plotted.
data = [10, 40, 50, 20, 10, 50]
worksheet.write_column('A1', data)
```

```
# Create a new chart object.
chart = workbook.add_chart({'type': 'line'})

# Add a series to the chart.
chart.add_series({'values': '=Sheet1!$A$1:$A$6'})

# Insert the chart into the worksheet.
worksheet.insert_chart('C1', chart)

workbook.close()
```

图表系列选项

以下部分详细介绍了`add_series()`Chart方法的更复杂选项：

```
marker
trendline
y_error_bars
x_error_bars
data_labels
points
smooth
```

图表系列选项：标记

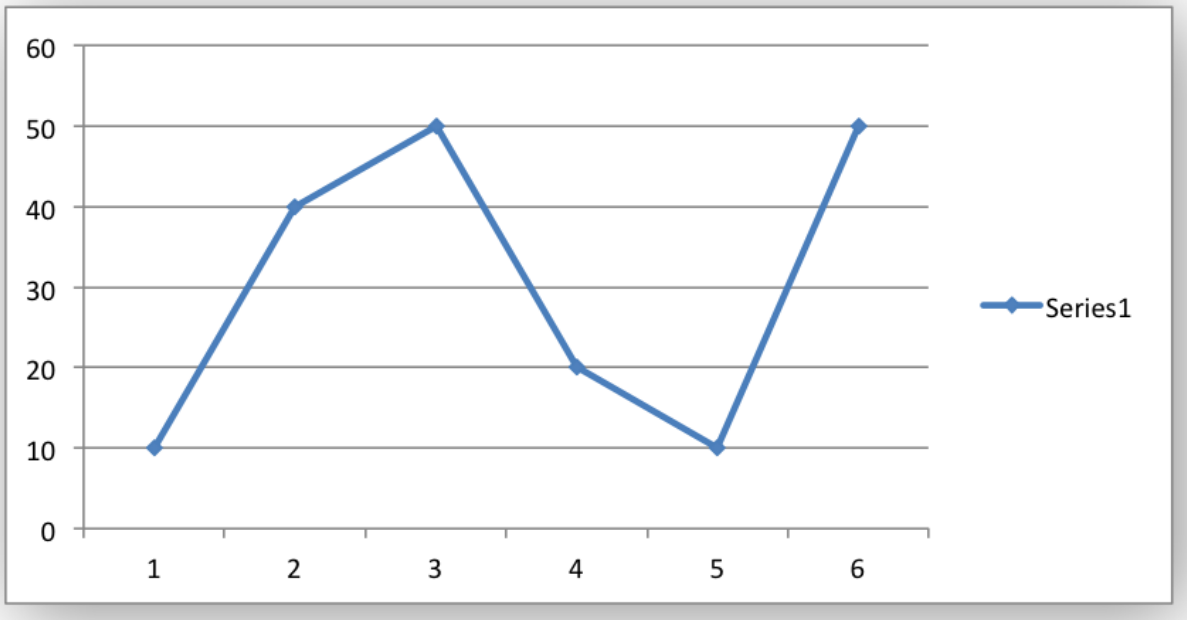
标记格式指定用于区分图表上系列的标记的属性。通常，只有Line和Scatter图表类型和趋势线使用标记。

可以为`marker`图表中的格式设置以下属性：

```
type
size
border
fill
pattern
gradient
```

该`type`属性设置与系列一起使用的标记类型：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'marker': {'type': 'diamond'},
})
```



`type`可以为`marker`图表中的格式设置以下属性。它们的显示顺序与Excel格式对话框中的顺序相同：

```
automatic
none
square
diamond
triangle
x
star
short_dash
long_dash
circle
plus
```

该`automatic`类型是一种特殊情况，它使用特定序列号的默认标记样式打开标记：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'marker': {'type': 'automatic'},
})
```

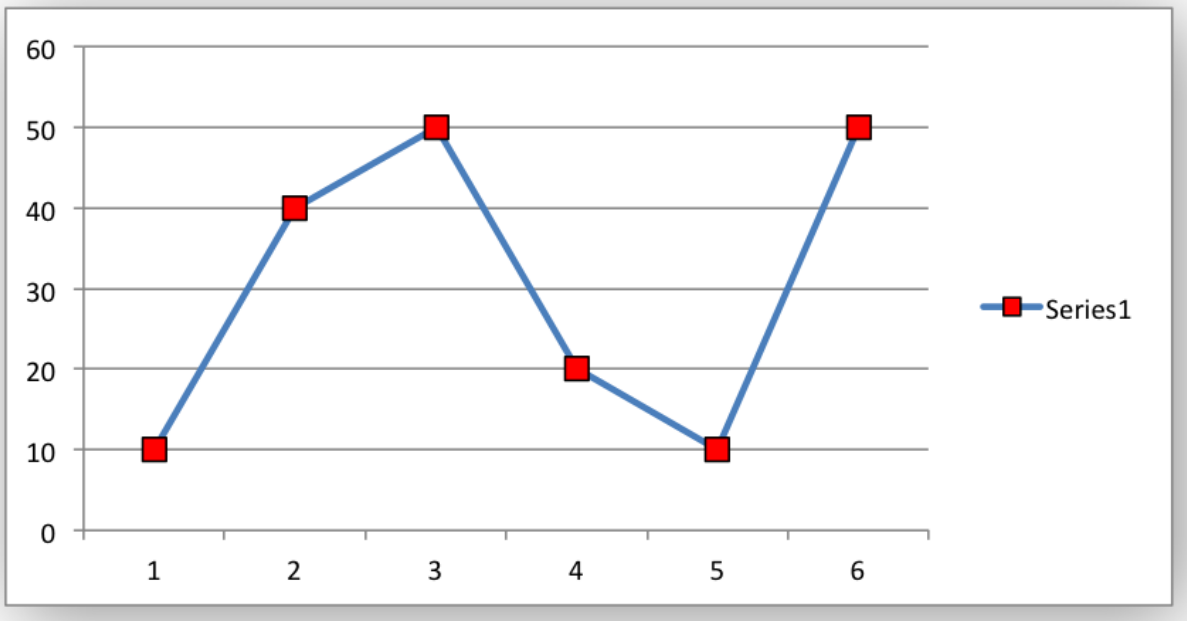
如果`automatic`处于启用状态，则无法设置其他标记属性，如大小，边框或填充。

该`size`属性设置标记的大小，通常与`type`以下内容一起使用：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'marker': {'type': 'diamond', 'size': 7},
})
```

也可以为标记设置嵌套`border`和`fill`属性：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'marker': {
        'type': 'square',
        'size': 8,
        'border': {'color': 'black'},
        'fill':   {'color': 'red'},
    },
})
```



图表系列选项：趋势线

可以将趋势线添加到图表系列中以指示数据中的趋势，例如移动平均值或多项式拟合。

可以为图表系列中的趋势线设置以下属性：

```
type
order                (for polynomial trends)
period               (for moving average)
forward              (for all except moving average)
backward             (for all except moving average)
name
line
intercept            (for exponential, linear and polynomial only)
display_equation     (for all except moving average)
display_r_squared    (for all except moving average)
```

该 `type` 属性设置系列中趋势线的类型：

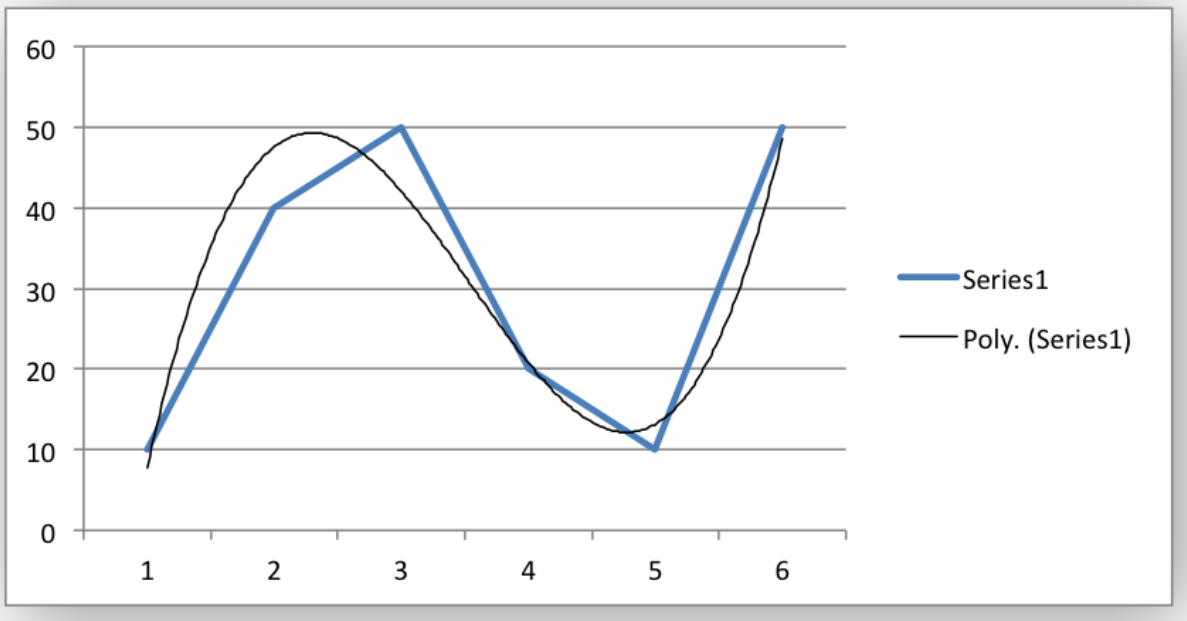
```
chart.add_series({
    'values':    '=Sheet1!$A$1:$A$6',
    'trendline': {'type': 'linear'},
})
```

可用的 `trendline` 类型是：

```
exponential
linear
log
moving_average
polynomial
power
```

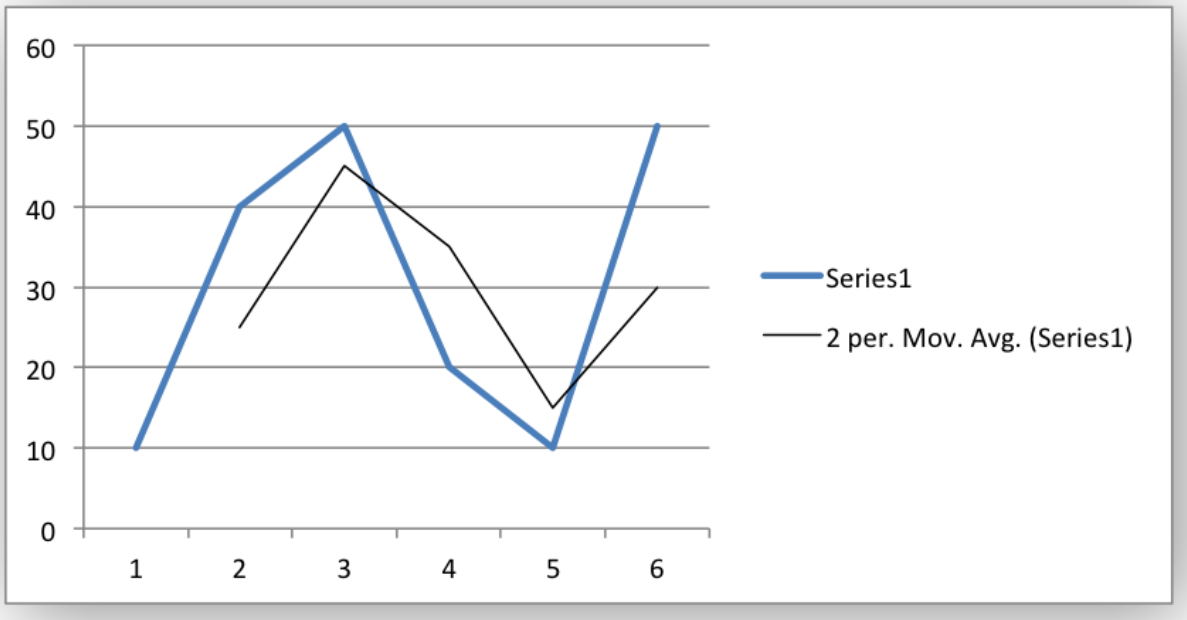
一个 `polynomial` 趋势线也可以指定 `order` 多项式的。默认值为2：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'trendline': {
        'type': 'polynomial',
        'order': 3,
    },
})
```



一个 `moving_average` 趋势线也可以指定 `period` 移动平均。默认值为2：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'trendline': {
        'type': 'moving_average',
        'period': 2,
    },
})
```



在`forward`和`backward`属性设置的趋势线的预测期：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'trendline': {
        'type': 'polynomial',
        'order': 2,
        'forward': 0.5,
        'backward': 0.5,
    },
})
```

该`name`属性为趋势线设置一个可选名称，该名称将显示在图表图例中。如果未指定，将显示Excel默认名称。这通常是趋势线类型和系列名称的组合：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'trendline': {
        'type': 'polynomial',
        'name': 'My trend name',
        'order': 2,
    },
})
```

该`intercept`属性设置趋势线与Y（值）轴相交的点：

```
chart.add_series({
    'values': '=Sheet1!$B$1:$B$5',
    'trendline': {'type': 'linear',
                  'intercept': 0.8,
    },
})
```

该`display_equation`属性在图表上显示趋势线方程：

```
chart.add_series({
    'values': '=Sheet1!$B$1:$B$5',
    'trendline': {'type': 'linear',
                  'display_equation': True,
    },
})
```

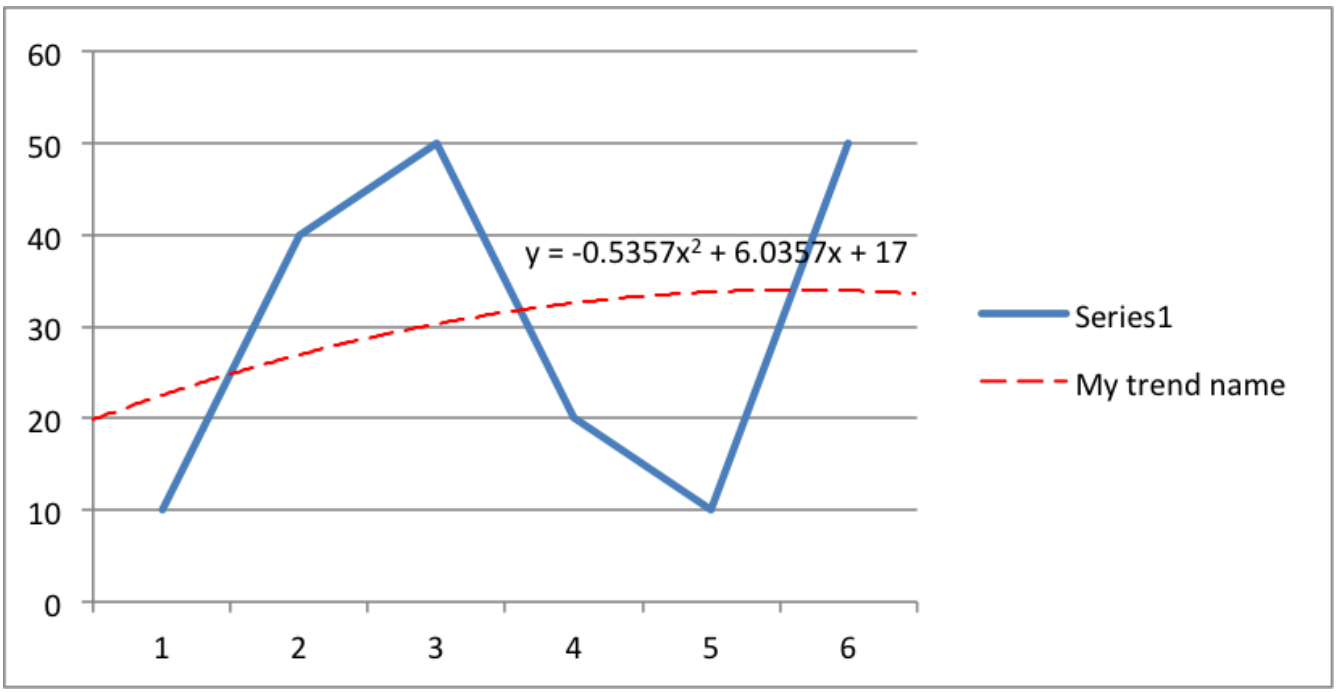
该`display_r_squared`属性显示图表上趋势线的R平方值：

```
chart.add_series({
    'values': '=Sheet1!$B$1:$B$5',
    'trendline': {'type': 'linear',
                  'display_r_squared': True,
    },
})
```

其中一些属性可以一次设置：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'trendline': {
        'type': 'polynomial',
        'name': 'My trend name',
        'order': 2,
        'forward': 0.5,
        'backward': 0.5,
        'display_equation': True,
        'line': {
            'color': 'red',
            'width': 1,
```

```
            'dash_type': 'long_dash',
        },
    },
})
```



趋势线无法添加到堆积图表或饼图，圆环图，雷达图表或（实施时）3D或曲面图表中的系列。

图表系列选项：误差条

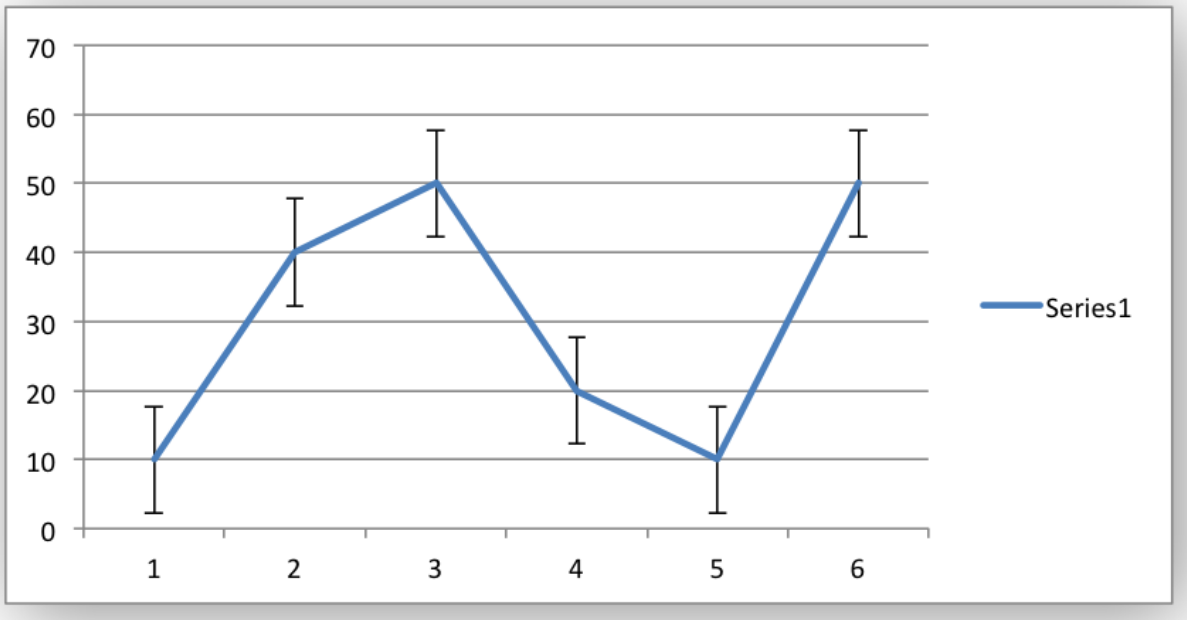
可以将错误栏添加到图表系列中以指示数据中的错误界限。误差线可以是垂直`y_error_bars`（最常见的类型）或水平`x_error_bars`（仅适用于Bar和Scatter图表）。

可以为图表系列中的误差线设置以下属性：

```
type
value        (for all types except standard error and custom)
plus_values  (for custom only)
minus_values (for custom only)
direction
end_style
line
```

该`type`属性设置系列中的错误栏类型：

```
chart.add_series({
    'values':       '=Sheet1!$A$1:$A$6',
    'y_error_bars': {'type': 'standard_error'},
})
```



可用的错误栏类型可用：

```
fixed
percentage
standard_deviation
standard_error
custom
```

所有错误栏类型，除了 `standard_error` 并且 `custom` 还必须具有与其关联的值以用于错误界限：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'y_error_bars': {
        'type': 'percentage',
        'value': 5,
    },
})
```

的 `custom` 误差棒型必须指定 `plus_values` 和 `minus_values` 其应当通过一个 `Sheet1!$A$1:$A$5` 类型范围式或值的列表：

```
chart.add_series({
    'categories': '=Sheet1!$A$1:$A$5',
    'values':     '=Sheet1!$B$1:$B$5',
    'y_error_bars': {
        'type':         'custom',
        'plus_values':  '=Sheet1!$C$1:$C$5',
        'minus_values': '=Sheet1!$D$1:$D$5',
    },
})
```

```
# or

chart.add_series({
    'categories': '=Sheet1!$A$1:$A$5',
    'values':     '=Sheet1!$B$1:$B$5',
    'y_error_bars': {
        'type':         'custom',
        'plus_values':  [1, 1, 1, 1, 1],
        'minus_values': [2, 2, 2, 2, 2],
    },
})
```

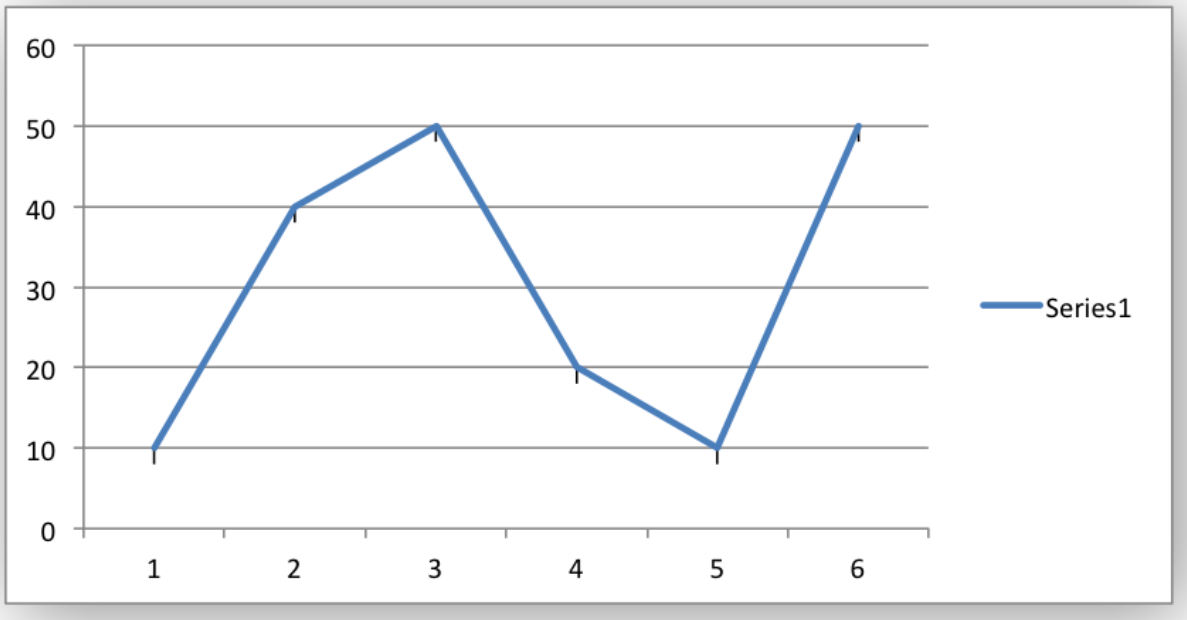
请注意，在Excel中，项目中的项目`minus_values`不需要为负数。

该`direction`属性设置误差线的方向。它应该是以下之一：

```
plus   # Positive direction only.
minus  # Negative direction only.
both   # Plus and minus directions, The default.
```

该`end_style`属性设置错误栏结束的样式。选项为1（默认值）或0（无端盖）：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'y_error_bars': {
        'type': 'fixed',
        'value': 2,
        'end_style': 0,
        'direction': 'minus'
    },
})
```



图表系列选项：数据标签

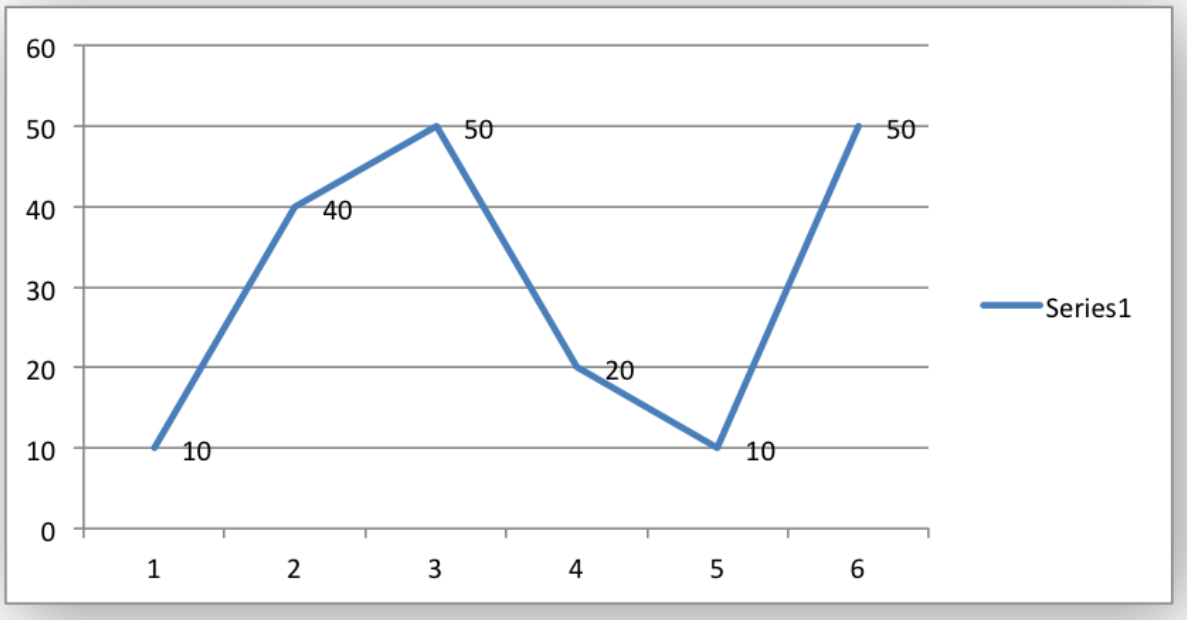
可以将数据标签添加到图表系列中以指示绘制的数据点的值。

可以为 `data_labels` 图表中的格式设置以下属性：

```
value
category
series_name
position
leader_lines
percentage
separator
legend_key
num_format
font
```

该 `value` 属性打开一个系列的*Value*数据标签：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'value': True},
})
```



该 `category` 属性打开系列的*类别名称*数据标签：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'category': True},
})
```

该 `series_name` 属性打开*系列的系列名称*数据标签：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'series_name': True},
})
```

该 `position` 属性用于定位系列的数据标签：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'series_name': True, 'position': 'center'},
})
```

在Excel中，允许的数据标签位置因不同的图表类型而异。允许的位置是：

位置	线，散点图，股票	酒吧，专栏	派，甜甜圈	区域，雷达
中央	是	是	是	是*
对	是*			
剩下	是			
以上	是			
下面	是			
inside_base		是		
inside_end		是	是	
outside_end		是*	是	
最合适			是*	

注意：*表示Excel中每种图表类型的默认位置（如果未指定位置）。

该`percentage`属性用于打开数据标签的显示，作为系列的 *百分比*。在Excel中，`percentage`数据标签选项仅适用于饼图和圆环图变体：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'percentage': True},
})
```

该`leader_lines`属性用于打开系列数据标签的*Leader Lines*。它主要用于饼图：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'value': True, 'leader_lines': True},
})
```

注意

即使打开引线，它们也不会在Excel或XlsxWriter中自动显示。由于Excel限制（或设计）引线仅在手动移动数据标签或数据标签非常接近且需要自动调整时才会出现。

该`separator`属性用于更改多个数据标签项之间的分隔符：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'value': True, 'category': True, 'separator': "\n"},
})
```

分隔符值必须是以下字符串之一：

```
','
';'
'.'
'\n'
' '
```

该`legend_key`属性用于打开系列数据标签的*Legend Key*：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$6',
    'data_labels': {'value': True, 'legend_key': True},
})
```

该`num_format`属性用于设置系列数据标签的数字格式：

```
chart.add_series({
    'values':      '=Sheet1!$A$1:$A$5',
    'data_labels': {'value': True, 'num_format': '#,##0.00'},
})
```

数字格式类似于工作表单元格格式`num_format`，但不能使用格式索引。必须使用显式格式字符串，如上所示。有关`set_num_format()`更多信息，请参阅

该`font`属性用于设置系列的数据标签的字体：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$5',
    'data_labels': {
        'value': True,
        'font': {'name': 'Consolas'}
    },
})
```

该`font`属性还用于旋转系列的数据标签：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$5',
    'data_labels': {
        'value': True,
        'font': {'rotation': 45}
    },
})
```

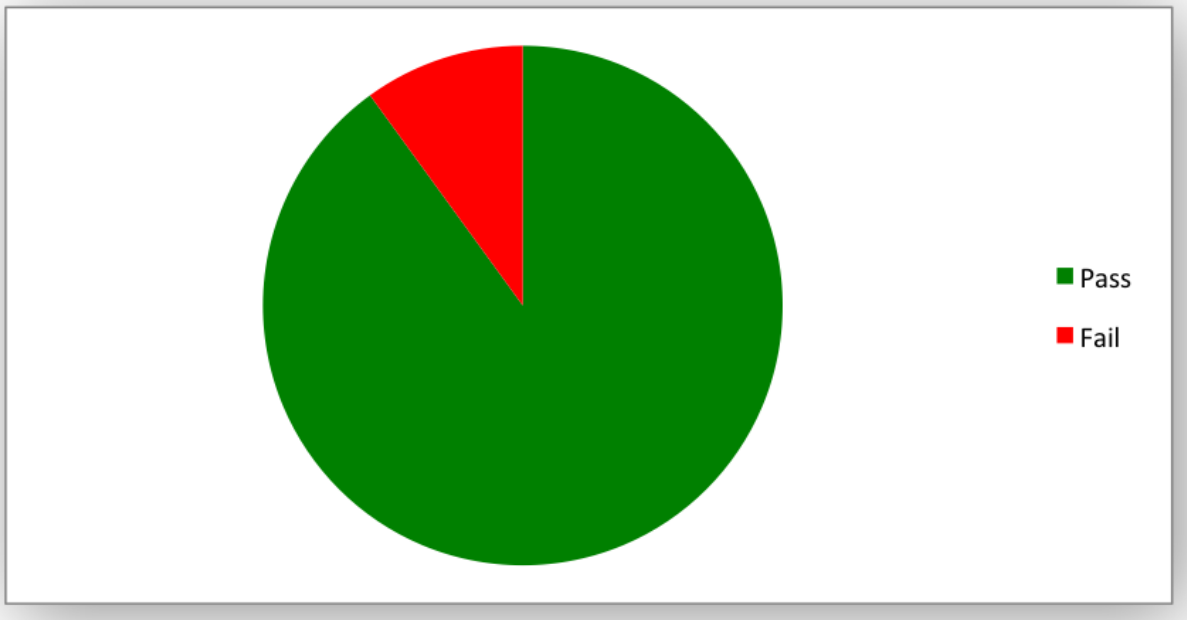
请参见图表字体。

图表系列选项：积分

通常，格式应用于图表中的整个系列。但是，有时需要对系列中的各个点进行格式化。特别是对于饼图/圆环图，这是必需的，其中每个片段由一个点表示。

在这些情况下，可以使用以下`points`属性 `add_series()`：

```
import xlsxwriter

workbook = xlsxwriter.Workbook('chart_pie.xlsx')

worksheet = workbook.add_worksheet()
chart = workbook.add_chart({'type': 'pie'})

data = [
    ['Pass', 'Fail'],
    [90, 10],
]

worksheet.write_column('A1', data[0])
worksheet.write_column('B1', data[1])

chart.add_series({
    'categories': '=Sheet1!$A$1:$A$2',
    'values':     '=Sheet1!$B$1:$B$2',
    'points': [
        {'fill': {'color': 'green'}},
        {'fill': {'color': 'red'}},
    ],
})

worksheet.insert_chart('C3', chart)

workbook.close()
```



该`points`属性采用格式选项列表（请参阅下面的“图表格式”部分）。要`None`为数组ref 中的系列传递值中的点指定默认属性：

```
# Format point 3 of 3 only.
chart.add_series({
    'values': '=Sheet1!A1:A3',
    'points': [
        None,
        None,
        {'fill': {'color': '#990000'}},
    ],
})

# Format point 1 of 3 only.
chart.add_series({
    'values': '=Sheet1!A1:A3',
    'points': [
        {'fill': {'color': '#990000'}},
    ],
})
```

图表系列选项：平滑

该`smooth`选项用于设置线系列的平滑属性。它仅适用于`line`和`scatter`图表类型：

```
chart.add_series({
    'categories': '=Sheet1!$A$1:$A$5',
    'values':     '=Sheet1!$B$1:$B$5',
    'smooth':     True,
})
```

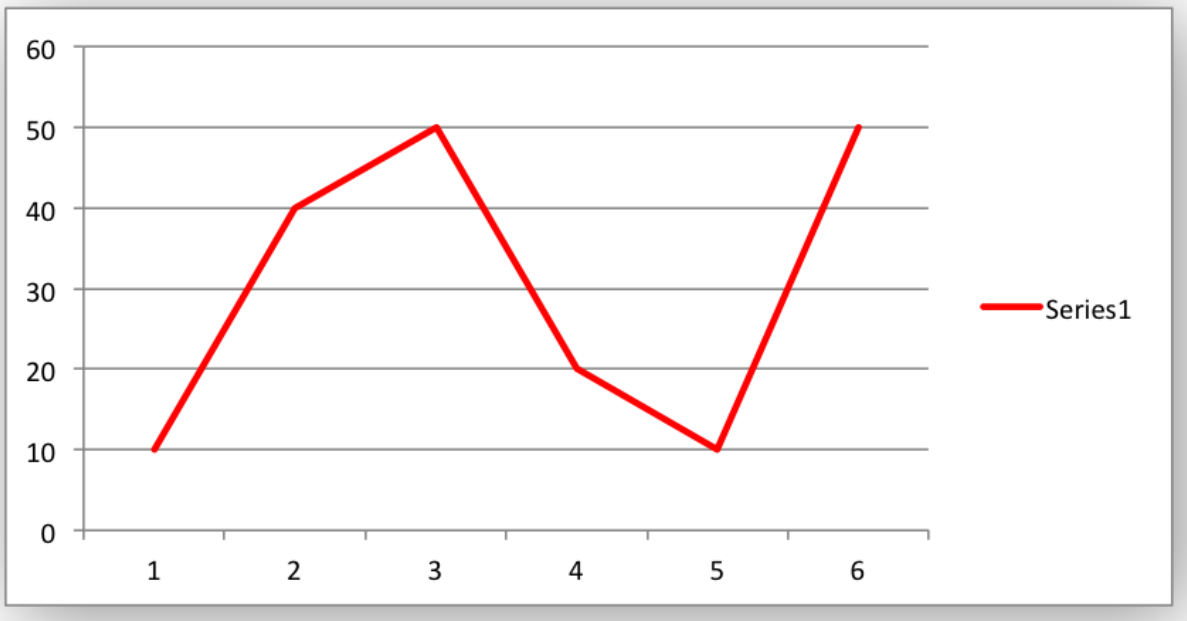
图表格式

可以为它们应用于（并且XlsxWriter支持的）任何图表对象设置以下图表格式设置属性，例如图表线，列填充区域，绘图区域边框，标记，网格线和其他图表元素：

```
line
border
fill
pattern
gradient
```

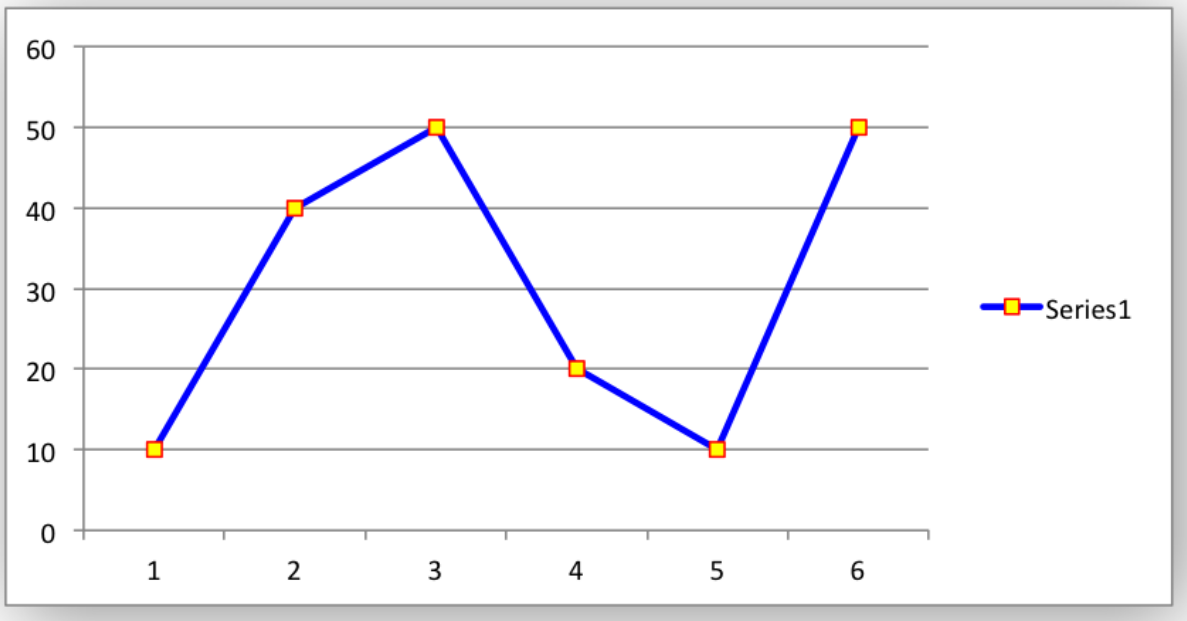
图表格式属性通常使用dicts设置：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'color': 'red'},
})
```



在某些情况下，格式属性可以嵌套。例如，a `marker` 可能包含 `border` 和 `fill` 子属性：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'color': 'blue'},
    'marker': {'type': 'square',
               'size,': 5,
               'border': {'color': 'red'},
               'fill':   {'color': 'yellow'}
    },
})
```



图表格式：行

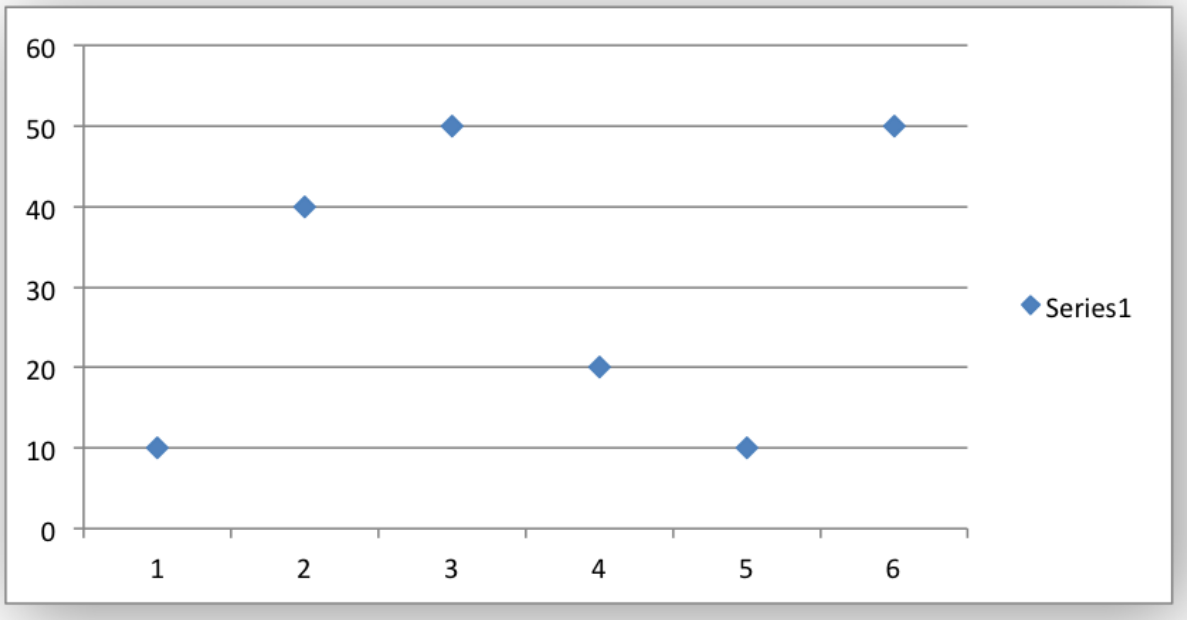
线格式用于指定图表中显示的线对象的属性，例如图表或边框上的绘制线。

可以为`line`图表中的格式设置以下属性：

```
none
color
width
dash_type
transparency
```

该`none`属性用于`line`关闭（默认情况下它始终打开，除了Scatter图表）。如果您希望绘制带有标记但没有线条的系列，这非常有用：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'none': True},
    'marker': {'type': 'automatic'},
})
```

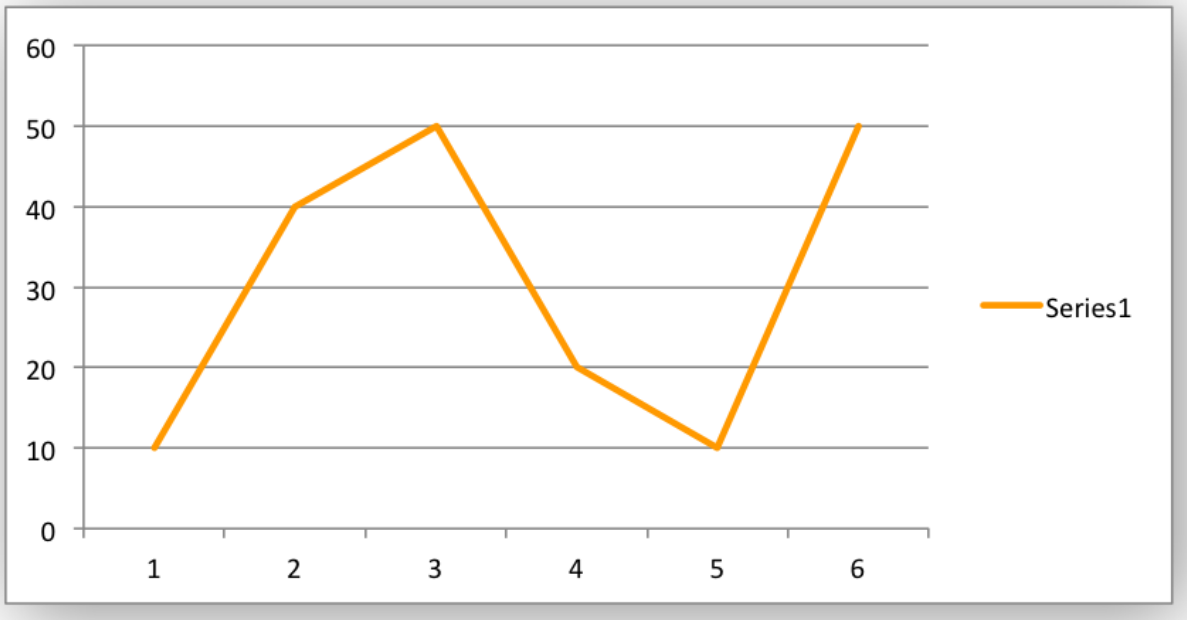


该`color`属性设置的颜色`line`：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'color': 'red'},
})
```

可用颜色显示在主XlsxWriter文档中。也可以使用Html样式`#RRGGBB`字符串或有限数量的命名颜色设置行的颜色，请参阅使用颜色：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'color': '#FF9900'},
})
```

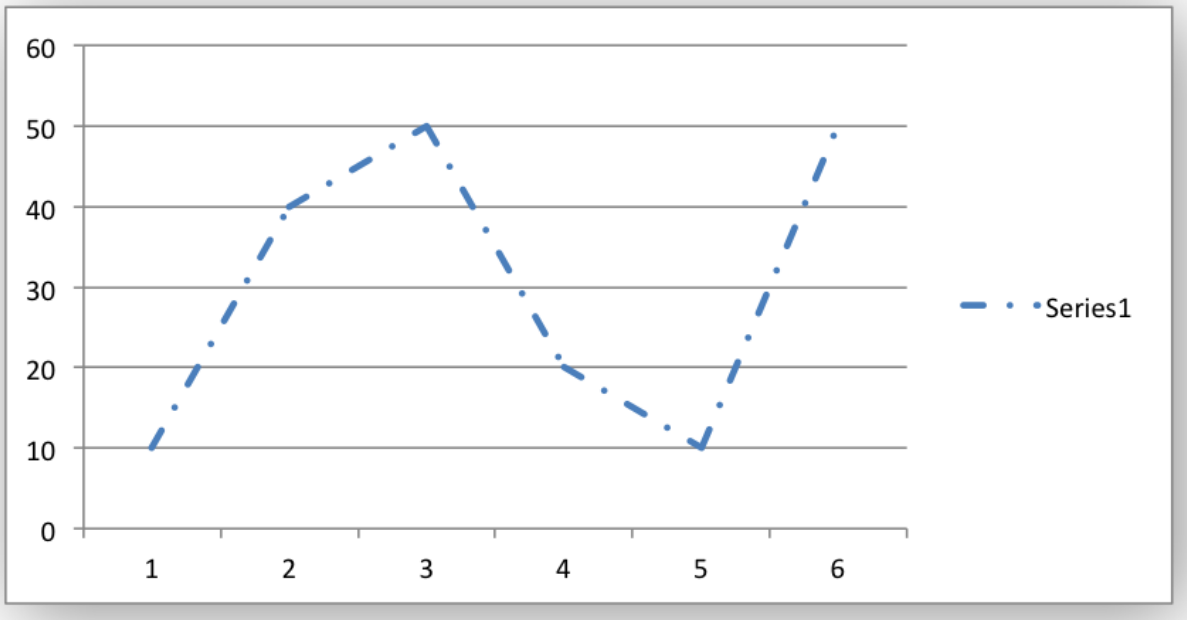


该`width`属性设置的宽度`line`。它应该以Excel中的0.25的增量指定：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'width': 3.25},
})
```

该`dash_type`属性设置该行的短划线样式：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'dash_type': 'dash_dot'},
})
```



可以使用以下`dash_type`值。它们按照它们出现在Excel对话框中的顺序显示：

```
solid
round_dot
square_dot
dash
dash_dot
long_dash
long_dash_dot
long_dash_dot_dot
```

默认的线条样式是`solid`。

该`transparency`属性将线条颜色的透明度设置为1到100的整数范围。必须设置颜色才能使透明度起作用，它不能使用自动/默认颜色：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'line':   {'color': 'yellow', 'transparency': 50},
})
```

一次`line`可以指定多个属性：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
     'line': {
         'color': 'red',
         'width': 1.25,
         'dash_type': 'square_dot',
     },
})
```

图表格式：边框

该`border`属性是同义词`line`。

它可以用作`line`图表类型的描述性替代，例如Bar和Column，它们具有边框和填充样式而不是线条样式。通常，具有`border`属性的图表对象也将具有填充属性。

图表格式：实心填充

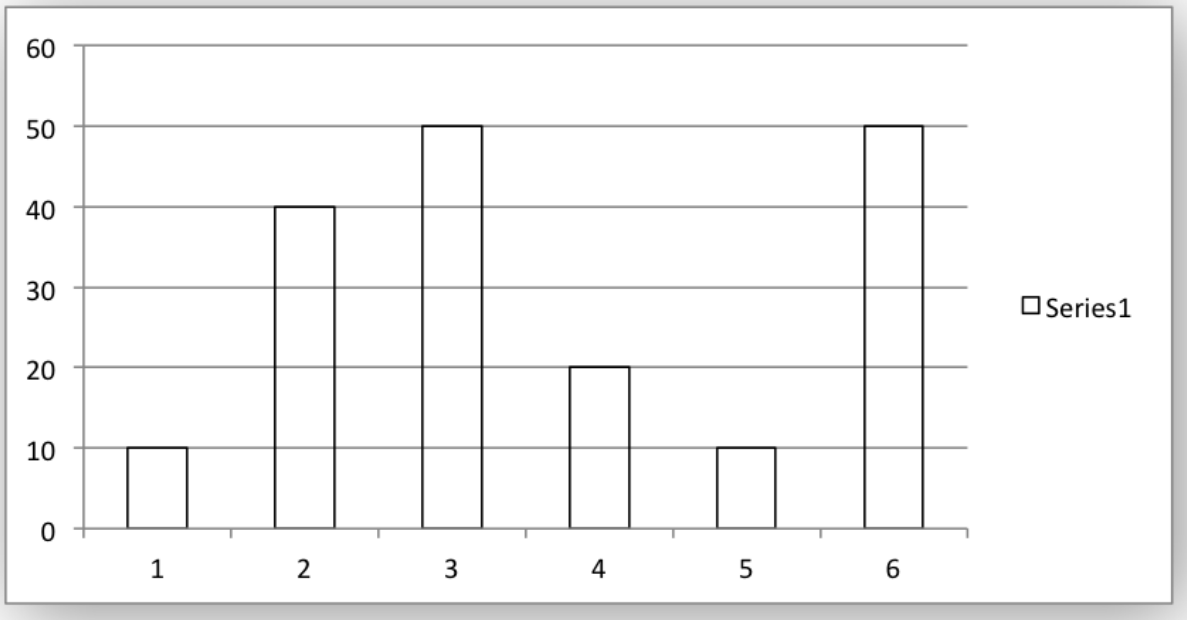
实心填充格式用于指定图表对象的填充区域，例如列的内部或图表本身的背景。

可以为`fill`图表中的格式设置以下属性：

```
none
color
transparency
```

该`none`属性用于关闭`fill`属性（默认情况下通常是打开）：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'fill':   {'none': True},
    'border': {'color': 'black'}
})
```

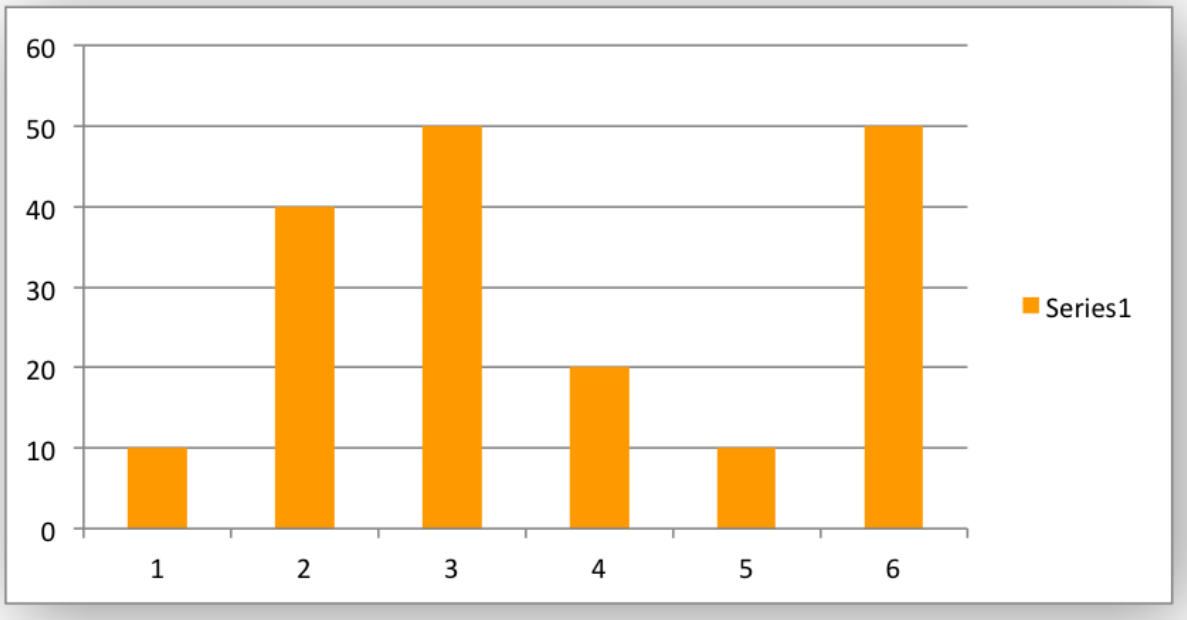


该`color`属性设置`fill`区域的颜色：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'fill':   {'color': 'red'}
})
```

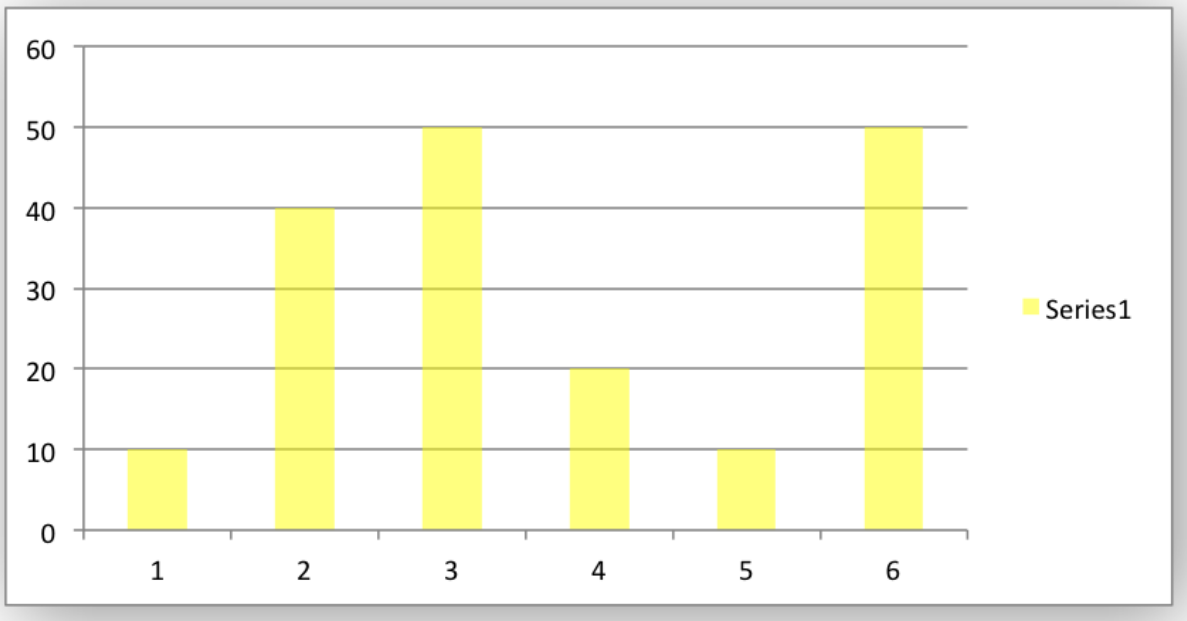
可用颜色显示在主XlsxWriter文档中。也可以使用Html样式`#RRGGBB`字符串或有限数量的命名颜色设置填充的颜色，请参阅使用颜色：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'fill':   {'color': '#FF9900'}
})
```



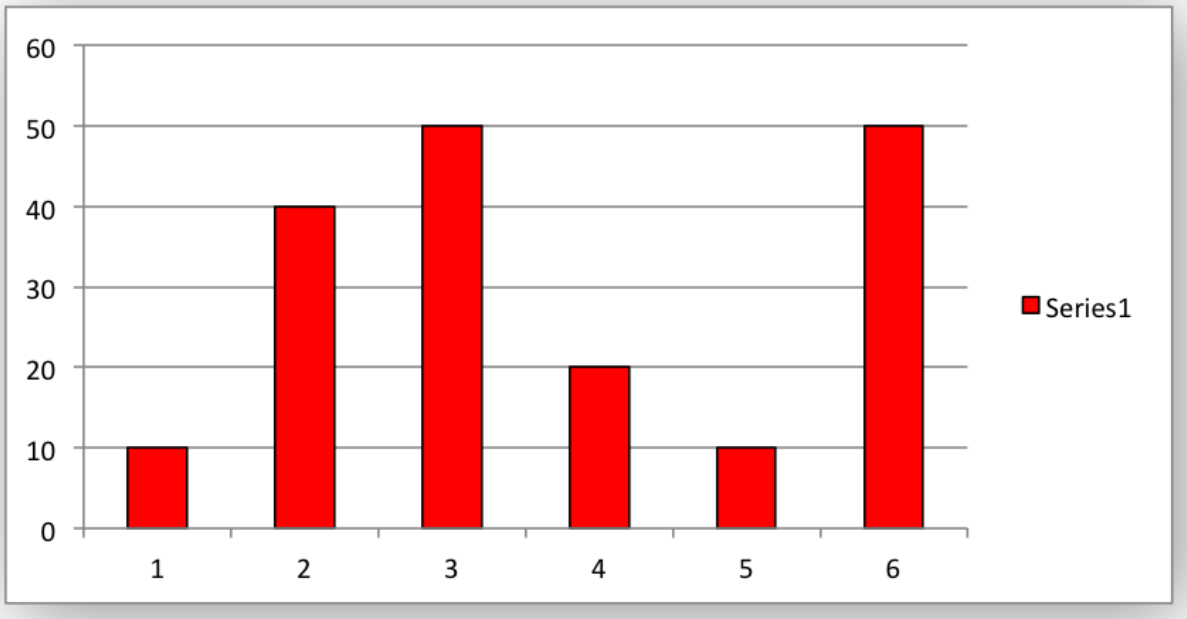
该`transparency`属性将整数填充颜色的透明度设置为1到100的整数。必须设置颜色才能使透明度起作用，它不适用于自动/默认颜色：

```
chart.set_chartarea({'fill': {'color': 'yellow', 'transparency': 50}})
```



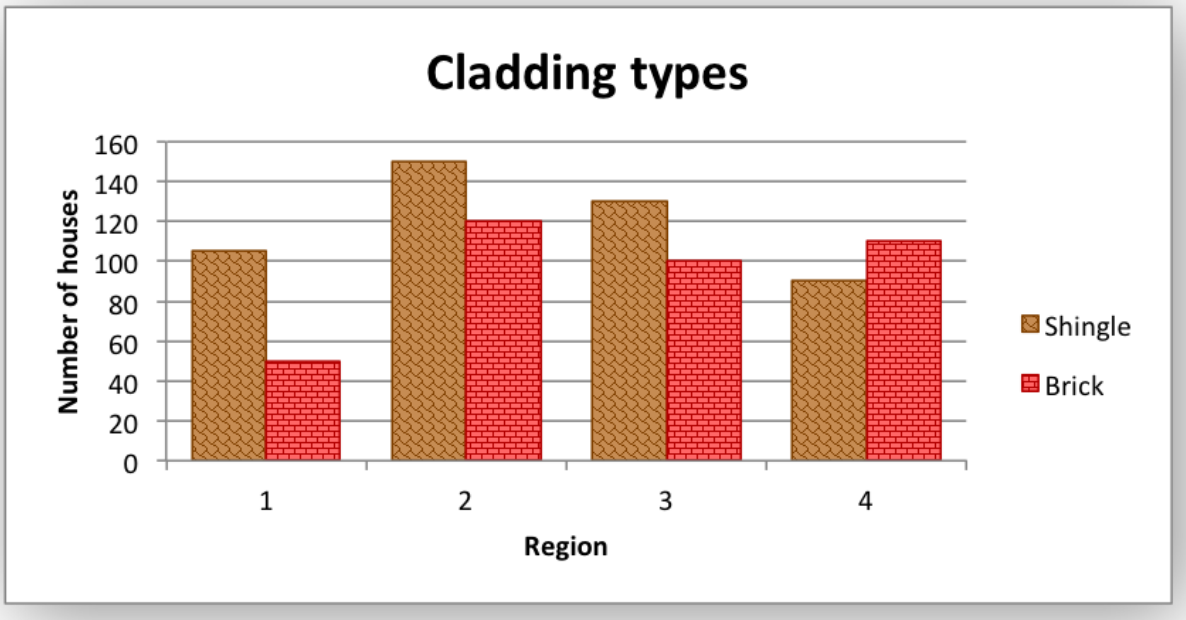
该`fill`格式通常用于在具有结合`border`其具有相同的特性的格式`line`的格式：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$6',
    'fill':   {'color': 'red'},
    'border': {'color': 'black'}
})
```



图表格式：图案填充

图案填充格式用于指定图表对象的图案填充区域，例如列的内部或图表本身的背景。



可以为 `pattern` 图表中的填充格式设置以下属性：

```
pattern:   the pattern to be applied (required)
fg_color:  the foreground color of the pattern (required)
bg_color:  the background color (optional, defaults to white)
```

例如：

```
chart.set_plotarea({
    'pattern': {
        'pattern': 'percent_5',
        'fg_color': 'red',
        'bg_color': 'yellow',
    }
})
```

可以应用以下模式：

- `percent_5`
- `percent_10`
- `percent_20`
- `percent_25`
- `percent_30`
- `percent_40`
- `percent_50`
- `percent_60`
- `percent_70`
- `percent_75`
- `percent_80`

- `percent_90`
- `light_downward_diagonal`
- `light_upward_diagonal`
- `dark_downward_diagonal`
- `dark_upward_diagonal`
- `wide_downward_diagonal`
- `wide_upward_diagonal`
- `light_vertical`
- `light_horizontal`
- `narrow_vertical`
- `narrow_horizontal`
- `dark_vertical`
- `dark_horizontal`
- `dashed_downward_diagonal`
- `dashed_upward_diagonal`
- `dashed_horizontal`
- `dashed_vertical`
- `small_confetti`
- `large_confetti`
- `zigzag`
- `wave`
- `diagonal_brick`
- `horizontal_brick`
- `weave`
- `plaid`
- `divot`
- `dotted_grid`
- `dotted_diamond`
- `shingle`
- `trellis`
- `sphere`
- `small_grid`
- `large_grid`
- `small_check`
- `large_check`
- `outlined_diamond`
- `solid_diamond`

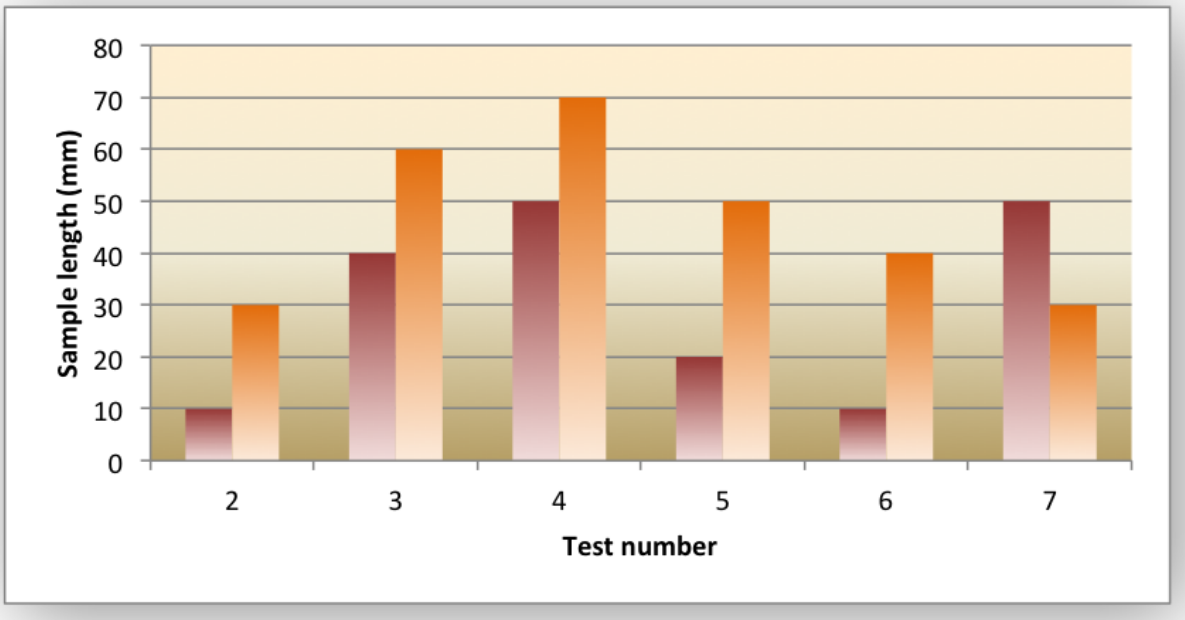
前景色 `fg_color` 是必需参数，可以是Html样式 `#RRGGBB` 字符串或有限数量的命名颜色，请参阅 使用颜色。

背景颜色 `bg_color` 是可选的，默认为白色。

如果在图表对象上使用图案填充，则它将覆盖对象的实心填充属性。

图表格式：渐变填充

渐变填充格式用于指定图表对象的渐变填充区域，例如列的内部或图表本身的背景。



可以为`gradient`图表中的填充格式设置以下属性：

```
colors:    a list of colors
positions: an optional list of positions for the colors
type:      the optional type of gradient fill
angle:     the optional angle of the linear fill
```

该`colors`属性设置一个定义以下内容的颜色列表`gradient`：

```
chart.set_plotarea({
    'gradient': {'colors': ['#FFEFD1', '#F0EBD5', '#B69F66']}
})
```

Excel允许在渐变中使用2到10种颜色，但您不太可能需要超过2或3种颜色。

与实心或图案填充一样，也可以使用Html样式`#RRGGBB`字符串或有限数量的命名颜色设置渐变的颜色，请参阅使用颜色：

```
chart.add_series({
    'values':   '=Sheet1!$A$1:$A$6',
    'gradient': {'colors': ['red', 'green']}
})
```

的`positions`限定位置的可选列表，0和100之间，这里在渐变中的颜色的位置。为`colors`2到4之间的列表提供了默认值，但如果需要，可以指定它们：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$5',
    'gradient': {
        'colors':    ['#DDEBCF', '#156B13'],
        'positions': [10,        90],
    }
})
```

该`type`属性可以具有以下值之一：

```
linear        (the default)
radial
rectangular
path
```

例如：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$5',
    'gradient': {
        'colors': ['#DDEBCF', '#9CB86E', '#156B13'],
        'type': 'radial'
    }
})
```

如果`type`未指定，则默认为`linear`。

对于`linear`填充，还可以指定渐变的角度：

```
chart.add_series({
    'values': '=Sheet1!$A$1:$A$5',
    'gradient': {'colors': ['#DDEBCF', '#9CB86E', '#156B13'],
                 'angle': 45}
})
```

默认角度为90度。

如果在图表对象上使用渐变填充，则它将覆盖对象的实心填充和图案填充属性。

图表字体

可以为它们应用于（并且XlsxWriter支持的）任何图表对象设置以下字体属性，例如图表标题，轴标签，轴编号和数据标签：

```
name
size
bold
italic
underline
rotation
color
```

这些属性对应于等效的Worksheet单元格格式对象属性。有关格式属性及其设置方式的更多详细信息，请参阅"格式类"部分。

以下说明可用的字体属性：

- `name`：设置字体名称：

  ```
  chart.set_x_axis({'num_font':  {'name': 'Arial'}})
  ```

- `size`：设置字体大小：

  ```
  chart.set_x_axis({'num_font':  {'name': 'Arial', 'size': 9}})
  ```

- `bold`：设置字体粗体属性：

  ```
  chart.set_x_axis({'num_font':  {'bold': True}})
  ```

- `italic`：设置字体斜体属性：

  ```
  chart.set_x_axis({'num_font':  {'italic': True}})
  ```

- `underline`：设置字体下划线属性：

  ```
  chart.set_x_axis({'num_font':  {'underline': True}})
  ```

- `rotation`：将字体旋转，角度，属性设置为-90到90度或270度的整数范围：

  ```
  chart.set_x_axis({'num_font':  {'rotation': 45}})
  ```

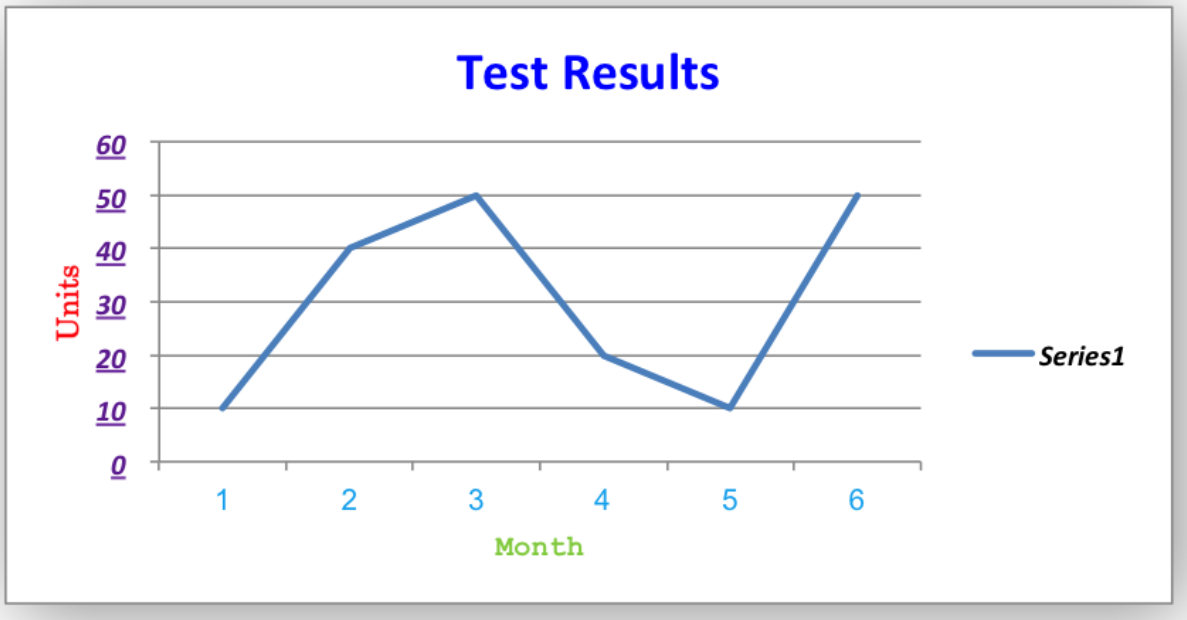
  字体旋转角度对于以更紧凑的格式显示轴数据（如日期）非常有用。还支持角度270。这表示字母从上到下（堆叠）的文本。

- `color`：设置字体颜色属性。可以是颜色索引，颜色名称或HTML样式RGB颜色：

  ```
  chart.set_x_axis({'num_font': {'color': 'red' }})
  chart.set_y_axis({'num_font': {'color': '#92D050'}})
  ```

以下是图表程序中字体格式的示例：

```
chart.set_title({
    'name': 'Test Results',
    'name_font': {
        'name': 'Calibri',
        'color': 'blue',
    },
})

chart.set_x_axis({
    'name': 'Month',
    'name_font': {
        'name': 'Courier New',
        'color': '#92D050'
    },
    'num_font': {
        'name': 'Arial',
        'color': '#00B0F0',
    },
})

chart.set_y_axis({
    'name': 'Units',
    'name_font': {
        'name': 'Century',
        'color': 'red'
    },
    'num_font': {
        'bold': True,
        'italic': True,
        'underline': True,
        'color': '#7030A0',
    },
})

chart.set_legend({'font': {'bold': 1, 'italic': 1}})
```



图表布局

工作表中图表的位置由该 `set_size()` 方法控制 。

也可以更改以下图表子对象的布局：

```
plotarea
legend
title
x_axis caption
y_axis caption
```

这里有些例子：

```
chart.set_plotarea({
    'layout': {
        'x':      0.13,
        'y':      0.26,
        'width':  0.73,
        'height': 0.57,
    }
})

chart.set_legend({
    'layout': {
        'x':      0.80,
        'y':      0.37,
        'width':  0.12,
```

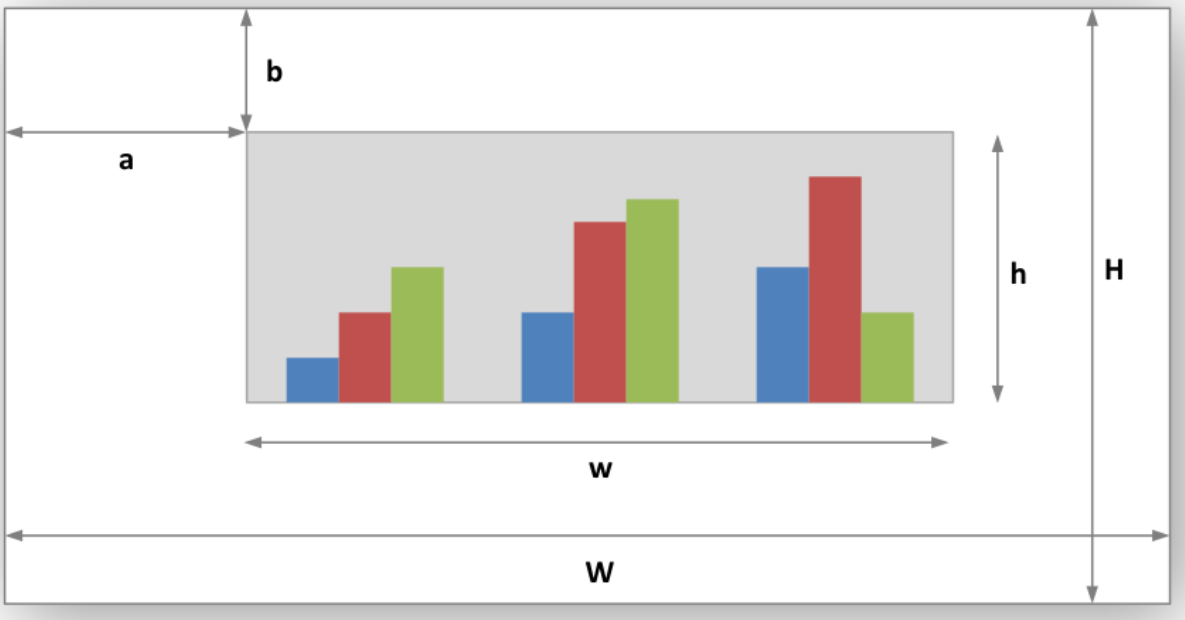
```
        'height': 0.25,
    }
})

chart.set_title({
    'name':    'Title',
    'overlay': True,
    'layout': {
        'x': 0.42,
        'y': 0.14,
    }
})

chart.set_x_axis({
    'name': 'X axis',
    'name_layout': {
        'x': 0.34,
        'y': 0.85,
    }
})
```

见`set_plotarea()`，`set_legend()`，`set_title()`和`set_x_axis()`，

注意

只能更改`plotarea`和`legend`对象的宽度和高度。对于其他基于文本的对象，宽度和高度由字体尺寸更改。

布局单位必须是范围内的浮点数，并表示为图表尺寸的百分比，如下所示：`0 < x <= 1`



由此布局单位计算如下：

```
layout:
    x      = a / W
    y      = b / H
    width  = w / W
    height = h / H
```

这些单位很麻烦，可能会因图表中的其他元素（如文本长度）而异。但是，这些是Excel允许相对定位所需的单位。通常需要一些反复试验。

注意

该`plotarea`原点在plotarea本身左上角，不考虑轴。

日期类别轴

日期类别轴是显示时间或日期信息的类别轴。在XlsxWriter中日期类别使用或中的`date_axis`选项 设置轴：`set_x_axis()` `set_y_axis()`

```
chart.set_x_axis({'date_axis': True})
```

通常，您还应该为日期轴指定数字格式，尽管Excel通常默认使用与绘制数据相同的格式：

```
chart.set_x_axis({
    'date_axis':  True,
    'num_format': 'dd/mm/yyyy',
})
```

Excel通常不允许为类别轴设置最小值和最大值。但是，日期轴是一个例外。该`min`和`max`值应设置为Excel时间或日期：

```
chart.set_x_axis({
    'date_axis': True,
    'min': date(2013, 1, 2),
    'max': date(2013, 1, 9),
    'num_format': 'dd/mm/yyyy',
})
```

对于日期轴，还可以设置主要和次要单位的类型：

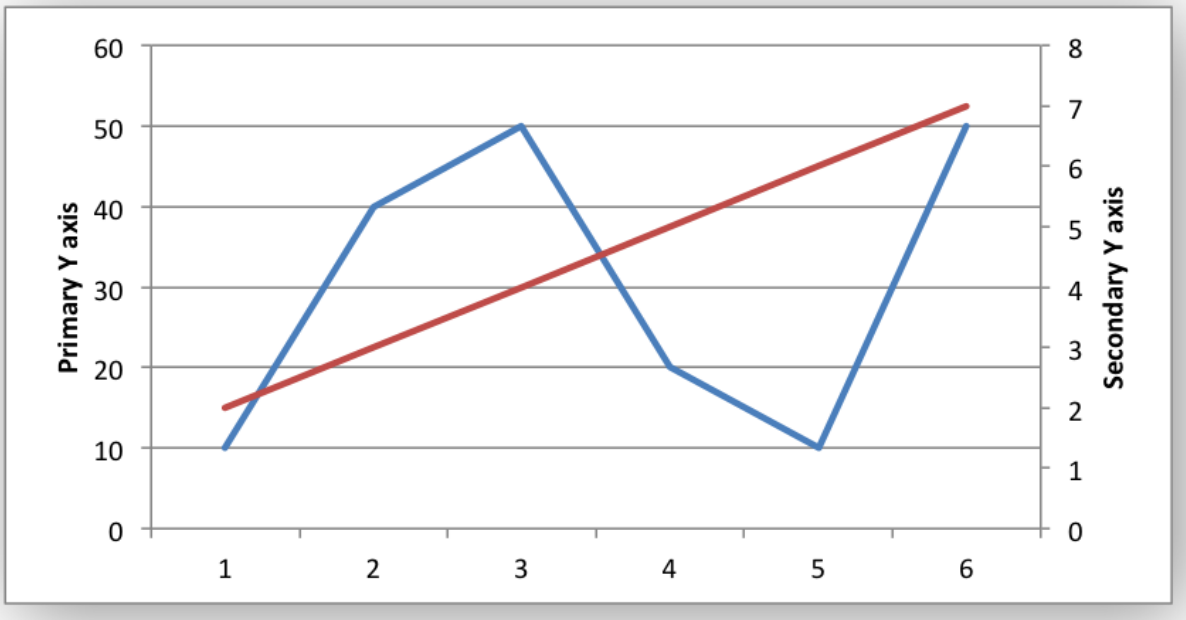
```
chart.set_x_axis({
    'date_axis':       True,
    'minor_unit':      4,
    'minor_unit_type': 'months',
    'major_unit':      1,
    'major_unit_type': 'years',
    'num_format':      'dd/mm/yyyy',
})
```

请参见示例：日期轴图表。

图表辅助轴

通过设置系列的`y2_axis`or `x2_axis`属性，可以将相同类型的辅助轴添加到图表中：

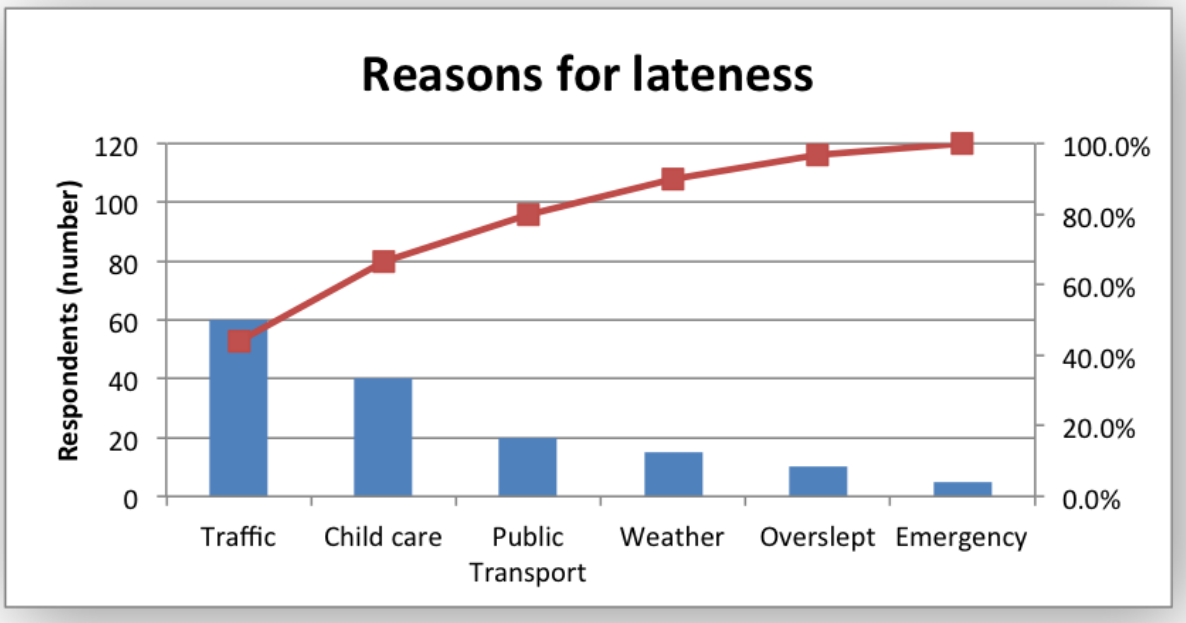
```
import xlsxwriter

workbook = xlsxwriter.Workbook('chart_secondary_axis.xlsx')
worksheet = workbook.add_worksheet()

data = [
    [2, 3, 4, 5, 6, 7],
    [10, 40, 50, 20, 10, 50],
]

worksheet.write_column('A2', data[0])
worksheet.write_column('B2', data[1])

chart = workbook.add_chart({'type': 'line'})

# Configure a series with a secondary axis.
chart.add_series({
    'values': '=Sheet1!$A$2:$A$7',
    'y2_axis': True,
})

# Configure a primary (default) Axis.
chart.add_series({
    'values': '=Sheet1!$B$2:$B$7',
})

chart.set_legend({'position': 'none'})

chart.set_y_axis({'name': 'Primary Y axis'})
chart.set_y2_axis({'name': 'Secondary Y axis'})

worksheet.insert_chart('D2', chart)

workbook.close()
```



也可以使用共享或辅助轴来绘制辅助组合图表，如下所示。

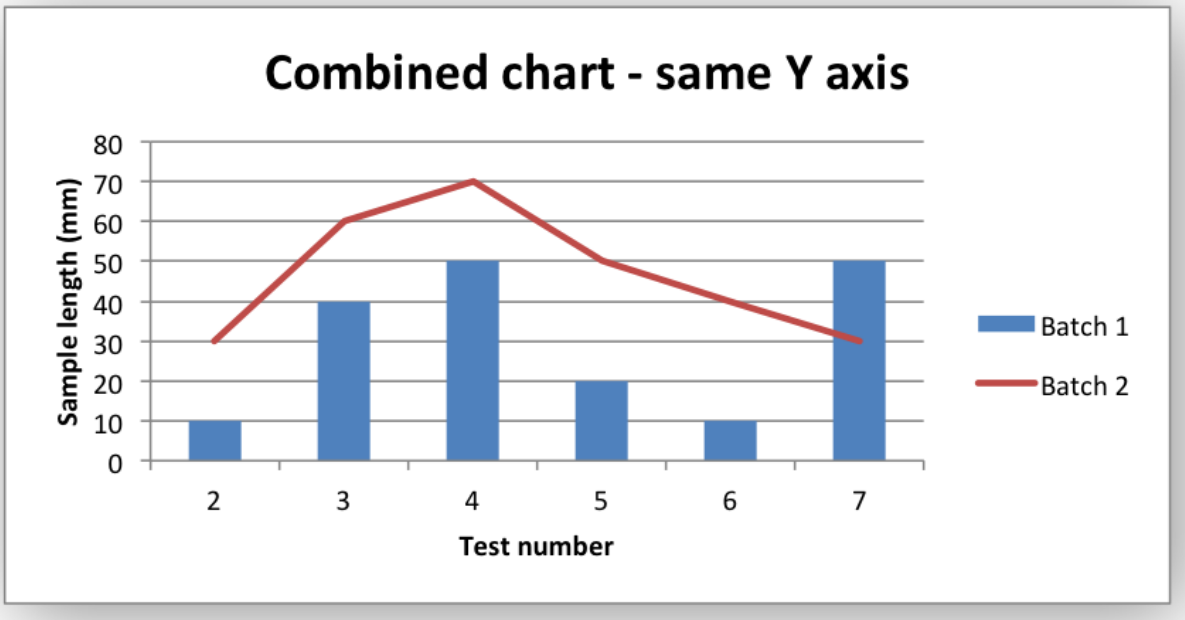
合并图表

也可以组合两种不同的图表类型，例如列和折线图，使用Chart `combine()` 方法创建Pareto图表：



组合图表可以共享相同的Y轴，如下例所示：

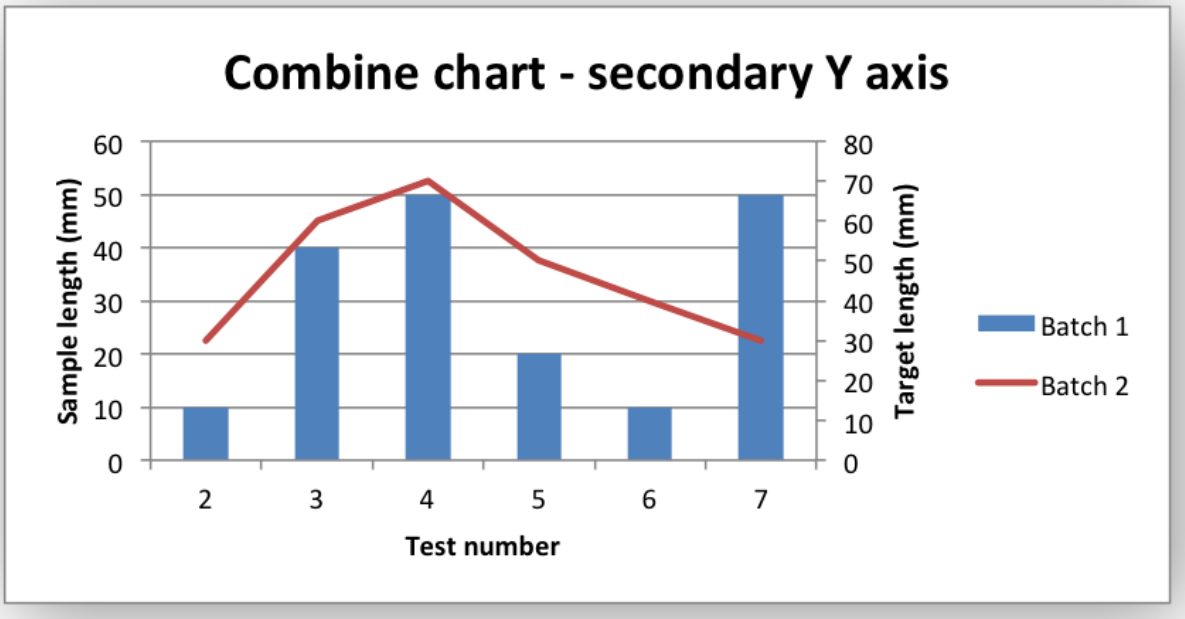
```
# Usual setup to create workbook and add data...

# Create a new column chart. This will use this as the primary chart.
column_chart = workbook.add_chart({'type': 'column'})

# Configure the data series for the primary chart.
column_chart.add_series({
    'name':       '=Sheet1!B1',
    'categories': '=Sheet1!A2:A7',
    'values':     '=Sheet1!B2:B7',
})

# Create a new column chart. This will use this as the secondary chart.
line_chart = workbook.add_chart({'type': 'line'})

# Configure the data series for the secondary chart.
line_chart.add_series({
    'name':       '=Sheet1!C1',
    'categories': '=Sheet1!A2:A7',
    'values':     '=Sheet1!C2:C7',
})

# Combine the charts.
column_chart.combine(line_chart)

# Add a chart title and some axis labels. Note, this is done via the
# primary chart.
column_chart.set_title({ 'name': 'Combined chart - same Y axis'})
column_chart.set_x_axis({'name': 'Test number'})
column_chart.set_y_axis({'name': 'Sample length (mm)'})

# Insert the chart into the worksheet
worksheet.insert_chart('E2', column_chart)
```



也可以使用上一节中显示的方法将辅助图表放置在辅助轴上。

在这种情况下，只需要 `y2_axis` 在系列中添加一个参数，如果需要，可以使用添加标题 `set_y2_axis()`。以下是上一个示例的附加内容，用于将辅助图表放在辅助轴上：

```
# ...
line_chart.add_series({
    'name':       '=Sheet1!C1',
    'categories': '=Sheet1!A2:A7',
    'values':     '=Sheet1!C2:C7',
    'y2_axis':    True,
})

# Add a chart title and some axis labels.
# ...
column_chart.set_y2_axis({'name': 'Target length (mm)'})
```



上面的示例使用*主要*和*次要*图表的概念。主图表是定义主X轴和Y轴的图表。它还用于设置除辅助数据系列之外的所有图表属性。例如，应通过主图表设置图表标题和轴属性。

另请参见示例：组合图表和示例：Pareto图表以获取更详细的示例。

组合图表有一些限制：

- 饼图目前无法合并。
- 散点图当前不能用作主图表，但可以用作辅助图表。
- 条形图只能在辅助轴上组合辅助图表。这是Excel限制。

工作表表格中的图表

可以从工作表表创建图表。但是，Excel有一个限制，即数据系列`name`（如果指定）必须引用表中的单元格（通常是其中一个标题）。

要解决此Excel限制，您可以在表中指定用户定义的名称，并从图表中引用该名称：

```
import xlsxwriter

workbook = xlsxwriter.Workbook('chart_pie.xlsx')

worksheet = workbook.add_worksheet()

data = [
    ['Apple',  60],
    ['Cherry', 30],
    ['Pecan',  10],
]

worksheet.add_table('A1:B4', {'data': data,
                              'columns': [{'header': 'Types'},
```

```
                                        {'header': 'Number'}]}
)

chart = workbook.add_chart({'type': 'pie'})

chart.add_series({
    'name':       '=Sheet1!$A$1',
    'categories': '=Sheet1!$A$2:$A$4',
    'values':     '=Sheet1!$B$2:$B$4',
})

worksheet.insert_chart('D2', chart)

workbook.close()
```

图表限制

XlsxWriter不支持以下图表功能：

- 3D图表和控件。
- “图表类”中未列出的“气泡”，“曲面”或其他图表类型。

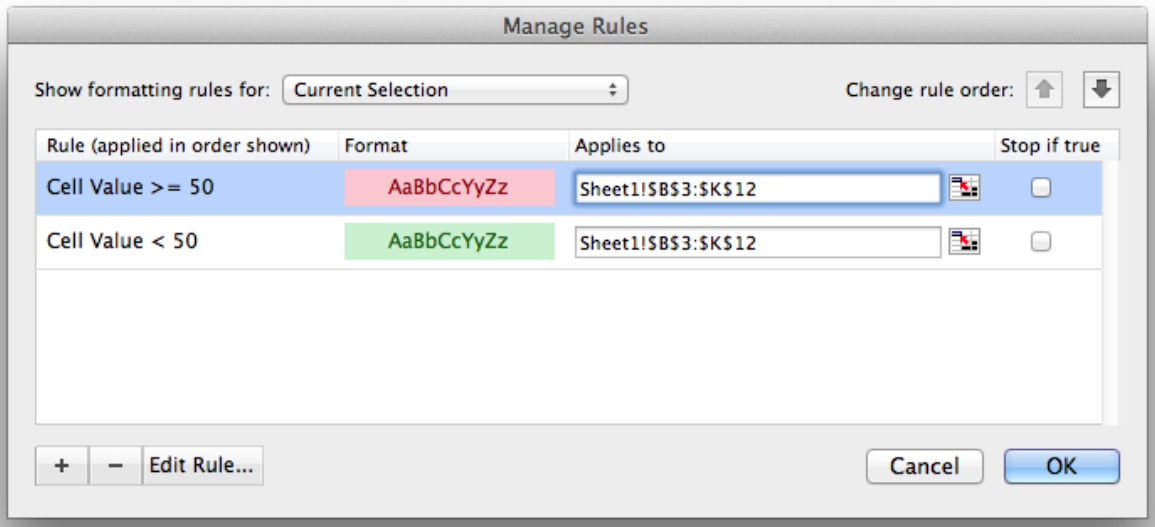
conditional format

条件格式是Excel的一项功能，允许您根据特定条件将格式应用于单元格或一系列单元格。

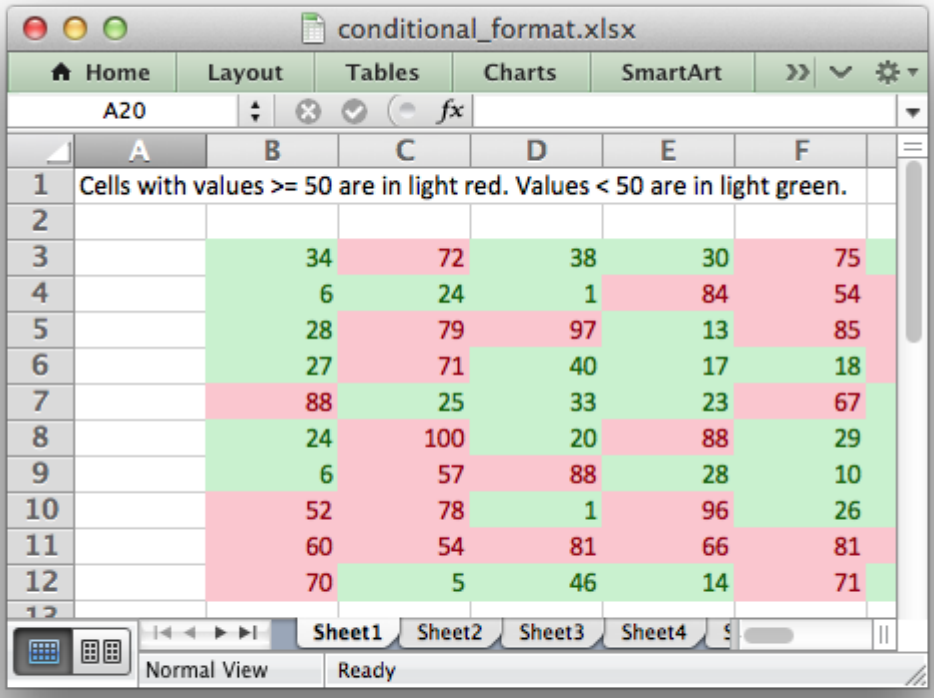
例如，以下规则用于突出显示conditional_format.py示例中的单元格：

```
worksheet.conditional_format('B3:K12', {'type':     'cell',
                                        'criteria': '>=',
                                        'value':    50,
                                        'format':   format1})

worksheet.conditional_format('B3:K12', {'type':     'cell',
                                        'criteria': '<',
                                        'value':    50,
                                        'format':   format2})
```

这给出了这样的标准：

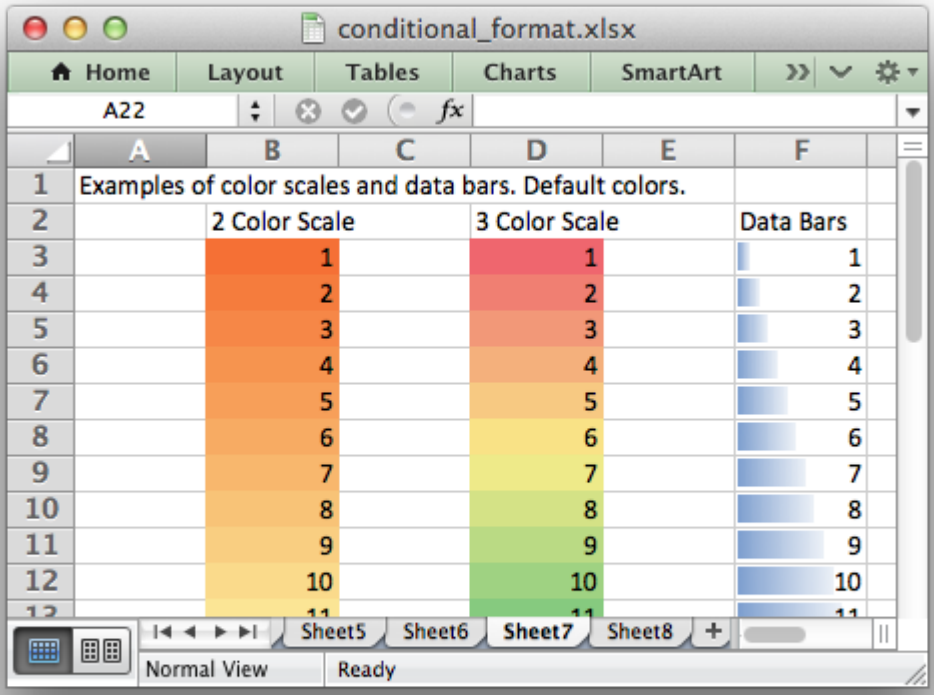


和输出看起来像这样：



	A	B	C	D	E	F
1	Cells with values >= 50 are in light red. Values < 50 are in light green.					
2						
3		34	72	38	30	75
4		6	24	1	84	54
5		28	79	97	13	85
6		27	71	40	17	18
7		88	25	33	23	67
8		24	100	20	88	29
9		6	57	88	28	10
10		52	78	1	96	26
11		60	54	81	66	81
12		70	5	46	14	71

也可以创建颜色标度和数据栏：



conditional_format（）方法

的 `conditional_format()` 工作表的方法使用基于用户定义的标准到XlsxWriter文件应用格式。

条件格式可以应用于单个单元格或一系列单元格。像往常一样，您可以使用A1或行/列表示法（使用单元格表示法）。

With Row/Column notation you must specify all four cells in the range: `(first_row, first_col, last_row, last_col)`. If you need to refer to a single cell set the `last_*` values equal to the `first_*` values. With A1 notation you can refer to a single cell or a range of cells:

```
worksheet.conditional_format(0, 0, 4, 1, {...})
worksheet.conditional_format('B1',       {...})
worksheet.conditional_format('C1:E5',    {...})
```

The options parameter in `conditional_format()` must be a dictionary containing the parameters that describe the type and style of the conditional format. The main parameters are:

- `type`
- `format`
- `criteria`
- `value`
- `minimum`
- `maximum`

Other, less commonly used parameters are:

- `min_type`
- `mid_type`
- `max_type`
- `min_value`
- `mid_value`
- `max_value`
- `min_color`
- `mid_color`
- `max_color`
- `bar_color`
- `bar_only`
- `bar_solid`
- `bar_negative_color`
- `bar_border_color`
- `bar_negative_border_color`
- `bar_negative_color_same`
- `bar_negative_border_color_same`
- `bar_no_border`
- `bar_direction`
- `bar_axis_position`
- `bar_axis_color`
- `data_bar_2010`
- `icon_style`
- `icons`
- `reverse_icons`
- `icons_only`
- `stop_if_true`
- `multi_range`

Conditional Format Options

The conditional format options that can be used with `conditional_format()` are explained in the following sections.

type

The `type` option is a required parameter and it has no default value. Allowable `type` values and their associated parameters are:

Type	Parameters
cell	criteria
	value
	minimum
	maximum
	format
date	criteria
	value
	minimum
	maximum
	format
time_period	criteria
	format
text	criteria
	value
	format
average	criteria
	format
duplicate	format
unique	format
top	criteria
	value
	format
bottom	criteria
	value
	format
blanks	format
no_blanks	format
errors	format

Type	Parameters
no_errors	format
formula	criteria
	format
2_color_scale	min_type
	max_type
	min_value
	max_value
	min_color
	max_color
3_color_scale	min_type
	mid_type
	max_type
	min_value
	mid_value
	max_value
	min_color
	mid_color
	max_color
data_bar	min_type
	max_type
	min_value
	max_value
	bar_only
	bar_color
	bar_solid*
	bar_negative_color*
	bar_border_color*
	bar_negative_border_color*

Type	Parameters
	bar_negative_color_same*
	bar_negative_border_color_same*
	bar_no_border*
	bar_direction*
	bar_axis_position*
	bar_axis_color*
	data_bar_2010*
icon_set	icon_style
	reverse_icons
	icons
	icons_only

Note

Data bar parameters marked with (*) are only available in Excel 2010 and later. Files that use these properties can still be opened in Excel 2007 but the data bars will be displayed without them.

type: cell

This is the most common conditional formatting type. It is used when a format is applied to a cell based on a simple criterion.

For example using a single cell and the `greater than` criteria:

```
worksheet.conditional_format('A1', {'type':     'cell',
                                    'criteria': 'greater than',
                                    'value':     5,
                                    'format':    red_format})
```

Or, using a range and the `between` criteria:

```
worksheet.conditional_format('C1:C4', {'type':     'cell',
                                       'criteria': 'between',
                                       'minimum':  20,
                                       'maximum':  30,
                                       'format':   green_format})
```

Other types are shown below, after the other main options.

criteria:

The `criteria` parameter is used to set the criteria by which the cell data will be evaluated. It has no default value. The most common criteria as applied to `{'type': 'cell'}` are:

`between`	
`not between`	
`equal to`	`==`
`not equal to`	`!=`
`greater than`	`>`
`less than`	`<`
`greater than or equal to`	`>=`
`less than or equal to`	`<=`

You can either use Excel's textual description strings, in the first column above, or the more common symbolic alternatives.

Additional criteria which are specific to other conditional format types are shown in the relevant sections below.

value:

The `value` is generally used along with the `criteria` parameter to set the rule by which the cell data will be evaluated:

```
worksheet.conditional_format('A1', {'type':     'cell',
                                    'criteria': 'equal to',
                                    'value':    5,
                                    'format':   red_format})
```

If the `type` is `cell` and the `value` is a string then it should be double quoted, as required by Excel:

```
worksheet.conditional_format('A1', {'type':     'cell',
                                    'criteria': 'equal to',
                                    'value':    '"Failed"',
                                    'format':   red_format})
```

The `value` property can also be an cell reference:

```
worksheet.conditional_format('A1', {'type':     'cell',
                                    'criteria': 'equal to',
                                    'value':    '$C$1',
                                    'format':   red_format})
```

Note

In general any `value` property that refers to a cell reference should use an absolute reference, especially if the conditional formatting is applied to a range of values. Without an absolute cell reference the conditional format will not be applied correctly by Excel from the first cell in the formatted range.

format:

The `format` parameter is used to specify the format that will be applied to the cell when the conditional formatting criterion is met. The format is created using the `add_format()` method in the same way as cell formats:

```
format1 = workbook.add_format({'bold': 1, 'italic': 1})

worksheet.conditional_format('A1', {'type':     'cell',
                                    'criteria': '>',
                                    'value':    5,
                                    'format':   format1})
```

Note

In Excel, a conditional format is superimposed over the existing cell format and not all cell format properties can be modified. Properties that **cannot** be modified in a conditional format are font name, font size, superscript and subscript, diagonal borders, all alignment properties and all protection properties.

Excel specifies some default formats to be used with conditional formatting. These can be replicated using the following XlsxWriter formats:

```
# Light red fill with dark red text.
format1 = workbook.add_format({'bg_color':   '#FFC7CE',
                               'font_color': '#9C0006'})

# Light yellow fill with dark yellow text.
format2 = workbook.add_format({'bg_color':   '#FFEB9C',
                               'font_color': '#9C6500'})

# Green fill with dark green text.
format3 = workbook.add_format({'bg_color':   '#C6EFCE',
                               'font_color': '#006100'})
```

See also The Format Class.

minimum:

The `minimum` parameter is used to set the lower limiting value when the `criteria` is either `'between'` or `'not between'`:

```
worksheet.conditional_format('A1', {'type':     'cell',
                                    'criteria': 'between',
                                    'minimum':  2,
                                    'maximum':  6,
                                    'format':   format1,
                                    })
```

maximum:

The `maximum` parameter is used to set the upper limiting value when the `criteria` is either `'between'` or `'not between'`. See the previous example.

type: date

The `date` type is similar the `cell` type and uses the same criteria and values. However, the `value`, `minimum` and `maximum` properties are specified as a datetime object as shown in Working with Dates and Time:

```
date = datetime.datetime.strptime('2011-01-01', "%Y-%m-%d")

worksheet.conditional_format('A1:A4', {'type':     'date',
                                       'criteria': 'greater than',
                                       'value':    date,
                                       'format':   format1})
```

type: time_period

The `time_period` type is used to specify Excel's "Dates Occurring" style conditional format:

```
worksheet.conditional_format('A1:A4', {'type':     'time_period',
                                       'criteria': 'yesterday',
                                       'format':   format1})
```

The period is set in the `criteria` and can have one of the following values:

```
'criteria': 'yesterday',
'criteria': 'today',
'criteria': 'last 7 days',
'criteria': 'last week',
'criteria': 'this week',
'criteria': 'next week',
'criteria': 'last month',
'criteria': 'this month',
'criteria': 'next month'
```

type: text

The `text` type is used to specify Excel's "Specific Text" style conditional format. It is used to do simple string matching using the `criteria` and `value` parameters:

```
worksheet.conditional_format('A1:A4', {'type':     'text',
                                       'criteria': 'containing',
                                       'value':    'foo',
                                       'format':   format1})
```

The `criteria` can have one of the following values:

```
'criteria': 'containing',
'criteria': 'not containing',
'criteria': 'begins with',
'criteria': 'ends with',
```

The `value` parameter should be a string or single character.

type: average

The `average` type is used to specify Excel's "Average" style conditional format:

```
worksheet.conditional_format('A1:A4', {'type':     'average',
                                       'criteria': 'above',
                                       'format':   format1})
```

The type of average for the conditional format range is specified by the `criteria`:

```
'criteria': 'above',
'criteria': 'below',
'criteria': 'equal or above',
'criteria': 'equal or below',
'criteria': '1 std dev above',
'criteria': '1 std dev below',
'criteria': '2 std dev above',
'criteria': '2 std dev below',
'criteria': '3 std dev above',
'criteria': '3 std dev below',
```

type: duplicate

The `duplicate` type is used to highlight duplicate cells in a range:

```
worksheet.conditional_format('A1:A4', {'type':   'duplicate',
                                       'format': format1})
```

type: unique

The `unique` type is used to highlight unique cells in a range:

```
worksheet.conditional_format('A1:A4', {'type':   'unique',
                                       'format': format1})
```

type: top

The `top` type is used to specify the top `n` values by number or percentage in a range:

```
worksheet.conditional_format('A1:A4', {'type':   'top',
                                       'value':  10,
                                       'format': format1})
```

The `criteria` can be used to indicate that a percentage condition is required:

```
worksheet.conditional_format('A1:A4', {'type':     'top',
                                       'value':    10,
                                       'criteria': '%',
                                       'format':   format1})
```

type: bottom

The `bottom` type is used to specify the bottom `n` values by number or percentage in a range.

It takes the same parameters as `top`, see above.

type: blanks

The `blanks` type is used to highlight blank cells in a range:

```
worksheet.conditional_format('A1:A4', {'type':   'blanks',
                                       'format': format1})
```

type: no_blanks

The `no_blanks` type is used to highlight non blank cells in a range:

```
worksheet.conditional_format('A1:A4', {'type':   'no_blanks',
                                       'format': format1})
```

type: errors

The `errors` type is used to highlight error cells in a range:

```
worksheet.conditional_format('A1:A4', {'type':   'errors',
                                       'format': format1})
```

type: no_errors

The `no_errors` type is used to highlight non error cells in a range:

```
worksheet.conditional_format('A1:A4', {'type':   'no_errors',
                                       'format': format1})
```

type: formula

The `formula` type is used to specify a conditional format based on a user defined formula:

```
worksheet.conditional_format('A1:A4', {'type':     'formula',
                                       'criteria': '=$A$1>5',
                                       'format':   format1})
```

The formula is specified in the `criteria`.

Formulas must be written with the US style separator/range operator which is a comma (not semi-colon) and should follow the same rules as `write_formula()`. See Non US Excel functions and syntax for a full explanation:

```
# This formula will cause an Excel error on load due to
# non-English language and use of semi-colons.
worksheet.conditional_format('A2:C9' ,
    {'type':     'formula',
     'criteria': '=ODER($B2<$C2;UND($B2="";$C2>HEUTE()))',
     'format':   format1
    })

# This is the correct syntax.
worksheet.conditional_format('A2:C9' ,
    {'type':     'formula',
     'criteria': '=OR($B2<$C2,AND($B2="",$C2>TODAY()))',
     'format':   format1
    })
```

Also, any cell or range references in the formula should be absolute references if they are applied to the full range of the conditional format. See the note in the `value` section above.

type: 2_color_scale

The `2_color_scale` type is used to specify Excel's "2 Color Scale" style conditional format:

```
worksheet.conditional_format('A1:A12', {'type': '2_color_scale'})
```

This conditional type can be modified with `min_type`, `max_type`, `min_value`, `max_value`, `min_color` and `max_color`, see below.

type: 3_color_scale

The `3_color_scale` type is used to specify Excel's "3 Color Scale" style conditional format:

```
worksheet.conditional_format('A1:A12', {'type': '3_color_scale'})
```

This conditional type can be modified with `min_type`, `mid_type`, `max_type`, `min_value`, `mid_value`, `max_value`, `min_color`, `mid_color` and `max_color`, see below.

type: data_bar

The `data_bar` type is used to specify Excel's "Data Bar" style conditional format:

```
worksheet.conditional_format('A1:A12', {'type': 'data_bar'})
```

This conditional type can be modified with the following parameters, which are explained in the sections below. These properties were available in the original xlsx file specification used in Excel 2007:

```
min_type
max_type
min_value
max_value
bar_color
bar_only
```

In Excel 2010 additional data bar properties were added such as solid (non-gradient) bars and control over how negative values are displayed. These properties can be set using the following parameters:

```
bar_solid
bar_negative_color
bar_border_color
bar_negative_border_color
bar_negative_color_same
bar_negative_border_color_same
bar_no_border
bar_direction
bar_axis_position
bar_axis_color
data_bar_2010
```

Files that use these Excel 2010 properties can still be opened in Excel 2007 but the data bars will be displayed without them.

type: icon_set

The `icon_set` type is used to specify a conditional format with a set of icons such as traffic lights or arrows:

```
worksheet.conditional_format('A1:C1', {'type': 'icon_set',
                                       'icon_style': '3_traffic_lights'})
```

The icon set style is specified by the `icon_style` parameter. Valid options are:

```
3_arrows
3_arrows_gray
3_flags
3_signs
3_symbols
3_symbols_circled
3_traffic_lights
```

```
3_traffic_lights_rimmed

4_arrows
4_arrows_gray
4_ratings
4_red_to_black
4_traffic_lights

5_arrows
5_arrows_gray
5_quarters
5_ratings
```

The criteria, type and value of each icon can be specified using the `icon` array of dicts with optional `criteria`, `type` and `value` parameters:

```
worksheet.conditional_format(
    'A1:D1',
    {'type': 'icon_set',
     'icon_style': '4_red_to_black',
     'icons': [{'criteria': '>=', 'type': 'number',     'value': 90},
               {'criteria': '<',  'type': 'percentile', 'value': 50},
               {'criteria': '<=', 'type': 'percent',    'value': 25}]}
)
```

- The icons `criteria` parameter should be either `>=` or `<`. The default `criteria` is `>=`.
- The icons `type` parameter should be one of the following values:

  ```
  number
  percentile
  percent
  formula
  ```

  The default `type` is `percent`.
- The icons `value` parameter can be a value or formula:

  ```
  worksheet.conditional_format('A1:D1',
                               {'type': 'icon_set',
                                'icon_style': '4_red_to_black',
                                'icons': [{'value': 90},
                                          {'value': 50},
                                          {'value': 25}]})
  ```

Note: The `icons` parameters should start with the highest value and with each subsequent one being lower. The default `value` is `(n * 100) / number_of_icons`. The lowest number icon in an icon set has properties defined by Excel. Therefore in a `n` icon set, there is no `n-1` hash of parameters.

The order of the icons can be reversed using the `reverse_icons` parameter:

```
worksheet.conditional_format('A1:C1',
                             {'type': 'icon_set',
                              'icon_style': '3_arrows',
                              'reverse_icons': True})
```

The icons can be displayed without the cell value using the `icons_only` parameter:

```
worksheet.conditional_format('A1:C1',
                             {'type': 'icon_set',
                              'icon_style': '3_flags',
                              'icons_only': True})
```

min_type:

The `min_type` and `max_type` properties are available when the conditional formatting type is `2_color_scale`, `3_color_scale` or `data_bar`. The `mid_type` is available for `3_color_scale`. The properties are used as follows:

```
worksheet.conditional_format('A1:A12', {'type':     '2_color_scale',
                                        'min_type': 'percent',
                                        'max_type': 'percent'})
```

The available min/mid/max types are:

```
min        (for min_type only)
num
percent
percentile
formula
max        (for max_type only)
```

mid_type:

Used for `3_color_scale`. Same as `min_type`, see above.

max_type:

Same as `min_type`, see above.

min_value:

The `min_value` and `max_value` properties are available when the conditional formatting type is `2_color_scale`, `3_color_scale` or `data_bar`. The `mid_value` is available for `3_color_scale`. The properties are used as follows:

```
worksheet.conditional_format('A1:A12', {'type':      '2_color_scale',
                                        'min_value': 10,
                                        'max_value': 90})
```

mid_value:

Used for `3_color_scale`. Same as `min_value`, see above.

max_value:

Same as `min_value`, see above.

min_color:

The `min_color` and `max_color` properties are available when the conditional formatting type is `2_color_scale`, `3_color_scale` or `data_bar`. The `mid_color` is available for `3_color_scale`. The properties are used as follows:

```
worksheet.conditional_format('A1:A12', {'type':      '2_color_scale',
                                        'min_color': '#C5D9F1',
                                        'max_color': '#538ED5'})
```

The color can be a Html style `#RRGGBB` string or a limited number named colors, see Working with Colors.

mid_color:

Used for `3_color_scale`. Same as `min_color`, see above.

max_color:

Same as `min_color`, see above.

bar_color:

The `bar_color` parameter sets the fill color for data bars:

```
worksheet.conditional_format('F3:F14', {'type': 'data_bar',
                                        'bar_color': '#63C384'})
```

The color can be a Html style `#RRGGBB` string or a limited number named colors, see Working with Colors.

bar_only:

The `bar_only` property displays a bar data but not the data in the cells:

```
worksheet.conditional_format('D3:D14', {'type': 'data_bar',
                                        'bar_only': True})
```

See the image above.

bar_solid:

The `bar_solid` property turns on a solid (non-gradient) fill for data bars:

```
worksheet.conditional_format('H3:H14', {'type': 'data_bar',
                                        'bar_solid': True})
```

See the image above.

Note, this property is only visible in Excel 2010 and later.

bar_negative_color:

The `bar_negative_color` property sets the color fill for the negative portion of a data bar:

```
worksheet.conditional_format('F3:F14', {'type': 'data_bar',
                                        'bar_negative_color': '#63C384'})
```

The color can be a Html style `#RRGGBB` string or a limited number named colors, see Working with Colors.

Note, this property is only visible in Excel 2010 and later.

bar_border_color:

The `bar_border_color` property sets the color for the border line of a data bar:

```
worksheet.conditional_format('F3:F14', {'type': 'data_bar',
                                        'bar_border_color': '#63C384'})
```

The color can be a Html style `#RRGGBB` string or a limited number named colors, see Working with Colors.

Note, this property is only visible in Excel 2010 and later.

bar_negative_border_color:

The `bar_negative_border_color` property sets the color for the border of the negative portion of a data bar:

```
worksheet.conditional_format('F3:F14', {'type': 'data_bar',
                                        'bar_negative_border_color': '#63C384'})
```

The color can be a Html style `#RRGGBB` string or a limited number named colors, see Working with Colors.

Note, this property is only visible in Excel 2010 and later.

bar_negative_color_same:

The `bar_negative_color_same` property sets the fill color for the negative portion of a data bar to be the same as the fill color for the positive portion of the data bar:

```
worksheet.conditional_format('F3:F14', {'type': 'data_bar',
                                        'bar_negative_color_same': True})
```

Note, this property is only visible in Excel 2010 and later.

bar_negative_border_color_same:

The `bar_negative_border_color_same` property sets the border color for the negative portion of a data bar to be the same as the border color for the positive portion of the data bar:

```
worksheet.conditional_format('F3:F14', {'type': 'data_bar',
                                        'bar_negative_border_color_same': True})
```

See the image above.

Note, this property is only visible in Excel 2010 and later.

bar_no_border:

The `bar_no_border` property turns off the border for data bars:

```
worksheet.conditional_format('F3:F14', {'type': 'data_bar',
                                        'bar_no_border': True})
```

Note, this property is only visible in Excel 2010 and later, however the default in Excel 2007 is to not have a border.

bar_direction:

The `bar_direction` property sets the direction for data bars. This property can be either `left` for left-to-right or `right` for right-to-left. If the property isn't set then Excel will adjust the position automatically based on the context:

```
worksheet.conditional_format('J3:J14', {'type': 'data_bar',
                                        'bar_direction': 'right'})
```

Note, this property is only visible in Excel 2010 and later.

bar_axis_position:

The `bar_axis_position` property sets the position within the cells for the axis that is shown in data bars when there are negative values to display. The property can be either `middle` or `none`. If the property isn't set then Excel will position the axis based on the range of positive and negative values:

```
worksheet.conditional_format('J3:J14', {'type': 'data_bar',
                                        'bar_axis_position': 'middle'})
```

Note, this property is only visible in Excel 2010 and later.

bar_axis_color:

The `bar_axis_color` property sets the color for the axis that is shown in data bars when there are negative values to display:

```
worksheet.conditional_format('J3:J14', {'type': 'data_bar',
                                        'bar_axis_color': '#0070C0'})
```

Note, this property is only visible in Excel 2010 and later.

data_bar_2010:

The `data_bar_2010` property sets Excel 2010 style data bars even when Excel 2010 specific properties aren't used. This can be used for consistency across all the data bar formatting in a worksheet:

```
worksheet.conditional_format('L3:L14', {'type': 'data_bar',
                                        'data_bar_2010': True})
```

stop_if_true

The `stop_if_true` parameter can be used to set the "stop if true" feature of a conditional formatting rule when more than one rule is applied to a cell or a range of cells. When this parameter is set then subsequent rules are not evaluated if the current rule is true:

```
worksheet.conditional_format('A1',
                             {'type': 'cell',
                              'format': cell_format,
                              'criteria': '>',
                              'value': 20,
                              'stop_if_true': True
                              })
```

multi_range:

The `multi_range` option is used to extend a conditional format over non-contiguous ranges.

可以使用多次调用将条件格式应用于工作表中的不同单元格范围`conditional_format()`。但是，作为次要优化，Excel中也可以将相同的条件格式应用于不同的非连续单元格范围。

这是在`conditional_format()`使用该`multi_range`选项时复制的。范围必须包含条件格式的主要范围以及由空格分隔的任何其他范围。

例如，将一个条件格式应用于两个范围，`'B3:K6'`并且`'B9:K12'`：

```
worksheet.conditional_format('B3:K6', {'type': 'cell',
                                       'criteria': '>=',
                                       'value': 50,
                                       'format': format1,
                                       'multi_range': 'B3:K6 B9:K12'})
```

条件格式示例

突出显示大于整数值的单元格：

```
worksheet.conditional_format('A1:F10', {'type':     'cell',
                                        'criteria': 'greater than',
                                        'value':    5,
                                        'format':   format1})
```

突出显示大于参考单元格中的值的单元格：

```
worksheet.conditional_format('A1:F10', {'type':     'cell',
                                        'criteria': 'greater than',
                                        'value':    'H1',
                                        'format':   format1})
```

突出显示比特定日期更近（更大）的单元格：

```
date = datetime.datetime.strptime('2011-01-01', "%Y-%m-%d")

worksheet.conditional_format('A1:F10', {'type':     'date',
                                        'criteria': 'greater than',
                                        'value':    date,
                                        'format':   format1})
```

突出显示过去七天中日期的单元格：

```
worksheet.conditional_format('A1:F10', {'type':     'time_period',
                                        'criteria': 'last 7 days',
                                        'format':   format1})
```

突出显示以字母开头的字符串`b`：

```
worksheet.conditional_format('A1:F10', {'type':     'text',
                                        'criteria': 'begins with',
                                        'value':    'b',
                                        'format':   format1})
```

突出显示高于该范围平均值1个标准差的单元格：

```
worksheet.conditional_format('A1:F10', {'type':   'average',
                                        'format': format1})
```

突出显示范围内的重复单元格：

```
worksheet.conditional_format('A1:F10', {'type':   'duplicate',
                                        'format': format1})
```

突出显示范围内的唯一单元格：

```
worksheet.conditional_format('A1:F10', {'type':   'unique',
                                        'format': format1})
```

突出显示前10个单元格：

```
worksheet.conditional_format('A1:F10', {'type':   'top',
                                        'value':  10,
                                        'format': format1})
```

突出显示空白单元格：

```
worksheet.conditional_format('A1:F10', {'type':   'blanks',
                                        'format': format1})
```

在3个单元格中设置交通灯图标：

```
worksheet.conditional_format('B3:D3', {'type': 'icon_set',
                                       'icon_style': '3_traffic_lights'})
```

add table detail

add_table（）

使用以下`add_table()`方法将表添加到工作表中：

```
worksheet.add_table('B3:F7', {options})
```

数据范围可以用“A1”或“行/列”表示法指定：

```
worksheet.add_table('B3:F7')
# Same as:
worksheet.add_table(2, 1, 6, 5)
```

The options parameter should be a dict containing the parameters that describe the table options and data. The available options are:

data
autofilter
header_row
banded_columns
banded_rows
first_column
last_column
style
total_row
columns
name

These options are explained below. There are no required parameters and the options parameter is itself optional if no options are specified (as shown above).

data

The `data` parameter can be used to specify the data in the cells of the table:

```
data = [
    ['Apples', 10000, 5000, 8000, 6000],
    ['Pears',   2000, 3000, 4000, 5000],
    ['Bananas', 6000, 6000, 6500, 6000],
    ['Oranges',  500,  300,  200,  700],

]

worksheet.add_table('B3:F7', {'data': data})
```

Table data can also be written separately, as an array or individual cells:

```
# These statements are the same as the single statement above.
worksheet.add_table('B3:F7')
worksheet.write_row('B4', data[0])
worksheet.write_row('B5', data[1])
worksheet.write_row('B6', data[2])
worksheet.write_row('B7', data[3])
```

Writing the cell data separately is occasionally required when you need to control the `write_()` methods used to populate the cells or if you wish to modify individual cell formatting.

The `data` structure should be an list of lists holding row data as shown above.

header_row

The `header_row` parameter can be used to turn on or off the header row in the table. It is on by default:

```
# Turn off the header row.
worksheet.add_table('B4:F7', {'header_row': False})
```

The header row will contain default captions such as `Column 1`, `Column 2`, etc. These captions can be overridden using the `columns` parameter below.

autofilter

The `autofilter` parameter can be used to turn on or off the autofilter in the header row. It is on by default:

```
# Turn off the default autofilter.
worksheet.add_table('B3:F7', {'autofilter': False})
```

The `autofilter` is only shown if the `header_row` is on. Filter conditions within the table are not supported.

banded_rows

The `banded_rows` parameter can be used to create rows of alternating color in the table. It is on by default:

```
# Turn off banded rows.
worksheet.add_table('B3:F7', {'banded_rows': False})
```

banded_columns

The `banded_columns` parameter can be used to used to create columns of alternating color in the table. It is off by default:

```
# Turn on banded columns.
worksheet.add_table('B3:F7', {'banded_columns': True})
```

See the above image.

first_column

The `first_column` parameter can be used to highlight the first column of the table. The type of highlighting will depend on the `style` of the table. It may be bold text or a different color. It is off by default:

```
# Turn on highlighting for the first column in the table.
worksheet.add_table('B3:F7', {'first_column': True})
```

last_column

The `last_column` parameter can be used to highlight the last column of the table. The type of highlighting will depend on the `style` of the table. It may be bold text or a different color. It is off by default:

```
# Turn on highlighting for the last column in the table.
worksheet.add_table('B3:F7', {'last_column': True})
```

See the above image.

style

The `style` parameter can be used to set the style of the table. Standard Excel table format names should be used (with matching capitalization):

```
worksheet.add_table('B3:F7', {'data': data,
                              'style': 'Table Style Light 11'})
```

The default table style is 'Table Style Medium 9'.

name

By default tables are named `Table1`, `Table2`, etc. The `name` parameter can be used to set the name of the table:

```
worksheet.add_table('B3:F7', {'name': 'SalesData'})
```

If you override the table name you must ensure that it doesn't clash with an existing table name and that it follows Excel's requirements for table names, see the Microsoft Office documentation.

If you need to know the name of the table, for example to use it in a formula, you can get it as follows:

```
table = worksheet.add_table('B3:F7')
table_name = table.name
```

total_row

The `total_row` parameter can be used to turn on the total row in the last row of a table. It is distinguished from the other rows by a different formatting and also with dropdown `SUBTOTAL` functions:

```
worksheet.add_table('B3:F7', {'total_row': True})
```

The default total row doesn't have any captions or functions. These must by specified via the `columns` parameter below.

columns

The `columns` parameter can be used to set properties for columns within the table.

The sub-properties that can be set are:

header
header_format
formula
total_string
total_function
总价值
格式

必须将列数据指定为dicts列表。例如，要覆盖默认的“Column n”样式表标题：

```
worksheet.add_table('B3:F7', {'data': data,
                              'columns': [{'header': 'Product'},
                                          {'header': 'Quarter 1'},
                                          {'header': 'Quarter 2'},
                                          {'header': 'Quarter 3'},
                                          {'header': 'Quarter 4'},
                                          ]})
```

请参见上面的结果图。

如果您不希望为特定列指定属性，则传递空哈希引用，并将应用默认值：

```
...
columns, [
    {header, 'Product'},
    {header, 'Quarter 1'},
    {},                     # Defaults to 'Column 3'.
    {header, 'Quarter 3'},
    {header, 'Quarter 4'},
]
...
```

可以使用column `formula`属性应用列公式：

```
formula = '=SUM(Table8[@[Quarter 1]:[Quarter 4]])'

worksheet.add_table('B3:G7', {'data': data,
                              'columns': [{'header': 'Product'},
                                          {'header': 'Quarter 1'},
                                          {'header': 'Quarter 2'},
                                          {'header': 'Quarter 3'},
                                          {'header': 'Quarter 4'},
                                          {'header': 'Year',
                                           'formula': formula},
                                          ]})
```

公式中支持Excel 2007样式和Excel 2010样式结构参考。但是，不支持对结构引用添加其他Excel 2010，并且公式应符合Excel 2007样式公式。有关 详细信息，请参阅有关对Excel表使用结构化引用的Microsoft文档 。`[#This Row]``@`

如上所述，`total_row`table参数打开表中的“Total”行，但不会使用任何默认值填充它。必须通过`columns`属性`total_string`和`total_function`子属性指定总标题和函数：

```
options = {'data': data,
           'total_row': 1,
           'columns': [{'header': 'Product', 'total_string': 'Totals'},
                       {'header': 'Quarter 1', 'total_function': 'sum'},
                       {'header': 'Quarter 2', 'total_function': 'sum'},
                       {'header': 'Quarter 3', 'total_function': 'sum'},
                       {'header': 'Quarter 4', 'total_function': 'sum'},
                       {'header': 'Year',
                        'formula': '=SUM(Table10[@[Quarter 1]:[Quarter 4]])',
                        'total_function': 'sum'
                        },
                       ]}

# Add a table to the worksheet.
worksheet.add_table('B3:G8', options)
```

支持的总计行`SUBTOTAL`功能是：

平均
count_nums
计数
最大
我
std_dev
和
是

不支持用户定义的函数或公式。

也可以为 `total_function` 使用 `total_value` 子属性设置计算值。仅在为无法自动计算公式值的应用程序创建工作簿时才需要这样做。这与在以下位置设置 `value` 可选属性 类似 `write_formula()` ：

```
options = {'data': data,
           'total_row': 1,
           'columns': [{'total_string': 'Totals'},
                       {'total_function': 'sum', 'total_value': 150},
                       {'total_function': 'sum', 'total_value': 200},
                       {'total_function': 'sum', 'total_value': 333},
                       {'total_function': 'sum', 'total_value': 124},
                       {'formula': '=SUM(Table10[@[Quarter 1]:[Quarter 4]])',
                        'total_function': 'sum',
                        'total_value': 807}]}
```

格式化也可以使用以下方式应用于列，使用 `format` 和应用于标题的列数据 `header_format` ：

```
currency_format = workbook.add_format({'num_format': '$#,##0'})
wrap_format     = workbook.add_format({'text_wrap': 1})

worksheet.add_table('B3:D8', {'data': data,
                              'total_row': 1,
                              'columns': [{'header': 'Product'},
                                          {'header': 'Quarter 1',
                                           'total_function': 'sum',
                                           'format': currency_format},
                                          {'header': 'Quarter 2',
                                           'header_format': wrap_format,
                                           'total_function': 'sum',
                                           'format': currency_format}]})
```

textbox detail

本节介绍如何使用XlsxWriter中文本框的一些选项和功能：

```
import xlsxwriter

workbook = xlsxwriter.Workbook('textbox.xlsx')
worksheet = workbook.add_worksheet()

text = 'Formatted textbox'

options = {
    'width': 256,
    'height': 100,
    'x_offset': 10,
    'y_offset': 10,

    'font': {'color': 'red',
             'size': 14},
    'align': {'vertical': 'middle',
              'horizontal': 'center'
              },
    'gradient': {'colors': ['#DDEBCF',
                            '#9CB86E',
                            '#156B13']},
}

worksheet.insert_textbox('B2', text, options)

workbook.close()
```

另请参见示例：将文本框插入工作表。

文本框选项

此Worksheet `insert_textbox()` 方法用于将文本框插入工作表：

```
worksheet.insert_textbox('B2', 'A simple textbox with some text')
```

文本可以包含换行文本的换行符：

```
worksheet.insert_textbox('B2', 'Line 1\nLine 2\n\nMore text')
```

这 `insert_textbox()` 需要一个可选 `dict` 参数，可用于控制文本框的大小，位置和格式：

```
worksheet.insert_textbox('B2', 'Some text', {'width': 256, 'height': 100})
```

可用选项包括：

```
# Size and position
width
height
x_scale
y_scale
```

```
x_offset
y_offset
object_position

# Formatting
line
border
fill
gradient
font
align
```

这些选项将在以下部分中介绍。它们与图表中使用的位置和格式参数类似或相同。

文本框大小和位置

`insert_textbox()` 控制文本框大小和位置的选项有：

```
width
height
x_scale
y_scale
x_offset
y_offset
object_position
```

The `width` and `height` are in pixels. The default textbox size is 192 x 120 pixels (or equivalent to 3 default columns x 6 default rows).

The size of the textbox can be modified by setting the `width` and `height` or by setting the `x_scale` and `y_scale`:

```
worksheet.insert_textbox('B2', 'Size adjusted textbox',
                         {'width': 288, 'height': 30})

# or ...
worksheet.insert_textbox('B2', 'Size adjusted textbox',
                         {'x_scale': 1.5, 'y_scale': 0.25})
```

The `x_offset` and `y_offset` position the top left corner of the textbox in the cell that it is inserted into.

The `object_position` parameter can be used to control the object positioning of the image:

```
worksheet.insert_textbox('B2', "Don't move or size with cells",
                         {'object_position': 3})
```

Where `object_position` has the following allowable values:

1. Move and size with cells (the default).
2. Move but don’t size with cells.
3. Don’t move or size with cells.

See Working with Object Positioning for more detailed information about the positioning and scaling of images within a worksheet.

Textbox Formatting

The following formatting properties can be set for textbox objects:

```
line
border
fill
gradient
font
align
```

Textbox formatting properties are set using the options dict:

```
worksheet.insert_textbox('B2', 'A textbox with a color text',
                                {'font': {'color': 'green'}})
```

In some cases the format properties can be nested:

```
worksheet.insert_textbox('B2', 'Some text in a textbox with formatting',
                         {'font': {'color': 'white'},
                          'align': {'vertical': 'middle',
                                    'horizontal': 'center'
                                    },
                          'gradient': {'colors': ['green', 'white']}})
```

Textbox formatting: Line

The line format is used to specify properties of the border in a textbox. The following properties can be set for `line` formats in a textbox:

```
none
color
width
dash_type
```

The `none` property is uses to turn the `line` off (it is always on by default):

```
worksheet.insert_textbox('B2', 'A textbox with no border line',
                         {'line': {'none': True}})
```

The `color` property sets the color of the `line`:

```
worksheet.insert_textbox('B2', 'A textbox with a color border',
                         {'line': {'color': 'red'}})
```

The available colors are shown in the main XlsxWriter documentation. It is also possible to set the color of a line with a Html style `#RRGGBB` string or a limited number of named colors, see Working with Colors:

```
worksheet.insert_textbox('B2', 'A textbox with a color border',
                         {'line': {'color': '#FF9900'}})
```

The `width` property sets the width of the `line`. It should be specified in increments of 0.25 of a point as in Excel:

```
worksheet.insert_textbox('B2', 'A textbox with larger border',
                         {'line': {'width': 3.25}})
```

The `dash_type` property sets the dash style of the line:

```
worksheet.insert_textbox('B2', 'A textbox a dash border',
                         {'line': {'dash_type': 'dash_dot'}})
```

The following `dash_type` values are available. They are shown in the order that they appear in the Excel dialog:

```
solid
round_dot
square_dot
dash
dash_dot
long_dash
long_dash_dot
long_dash_dot_dot
```

The default line style is `solid`.

More than one `line` property can be specified at a time:

```
worksheet.insert_textbox('B2', 'A textbox with border formatting',
                         {'line': {'color': 'red',
                                   'width': 1.25,
                                   'dash_type': 'square_dot'}})
```

Textbox formatting: Border

The `border` property is a synonym for `line`.

Excel uses a common dialog for setting object formatting but depending on context it may refer to a *line* or a *border*. For formatting these can be used interchangeably.

Textbox formatting: Solid Fill

The solid fill format is used to specify a fill for a textbox object.

The following properties can be set for `fill` formats in a textbox:

```
none
color
```

The `none` property is used to turn the `fill` property off (to make the textbox transparent):

```
worksheet.insert_textbox('B2', 'A textbox with no fill',
                         {'fill': {'none': True}})
```

The `color` property sets the color of the `fill` area:

```
worksheet.insert_textbox('B2', 'A textbox with color fill',
                         {'fill': {'color': '#FF9900'}})
```

The available colors are shown in the main XlsxWriter documentation. It is also possible to set the color of a fill with a Html style `#RRGGBB` string or a limited number of named colors, see Working with Colors:

```
worksheet.insert_textbox('B2', 'A textbox with color fill',
                         {'fill': {'color': 'red'}})
```

Textbox formatting: Gradient Fill

The gradient fill format is used to specify a gradient fill for a textbox. The following properties can be set for `gradient` fill formats in a textbox:

```
colors:    a list of colors
positions: an optional list of positions for the colors
type:      the optional type of gradient fill
angle:     the optional angle of the linear fill
```

If gradient fill is used on a textbox object it overrides the solid fill properties of the object.

The `colors` property sets a list of colors that define the `gradient`:

```
worksheet.insert_textbox('B2', 'A textbox with gradient fill',
                         {'gradient': {'colors': ['gray', 'white']}})
```

Excel allows between 2 and 10 colors in a gradient but it is unlikely that you will require more than 2 or 3.

As with solid fill it is also possible to set the colors of a gradient with a Html style `#RRGGBB` string or a limited number of named colors, see Working with Colors:

```
worksheet.insert_textbox('B2', 'A textbox with gradient fill',
                         {'gradient': {'colors': ['#DDEBCF',
                                                  '#9CB86E',
                                                  '#156B13']}})
```

The `positions` defines an optional list of positions, between 0 and 100, of where the colors in the gradient are located. Default values are provided for `colors` lists of between 2 and 4 but they can be specified if required:

```
worksheet.insert_textbox('B2', 'A textbox with gradient fill',
                         {'gradient': {'colors':    ['#DDEBCF', '#156B13'],
                                       'positions': [10,        90]}})
```

The `type` property can have one of the following values:

```
linear        (the default)
radial
rectangular
path
```

For example:

```
worksheet.insert_textbox('B2', 'A textbox with gradient fill',
                         {'gradient': {'colors': ['#DDEBCF', '#9CB86E', '#156B13'],
                                       'type': 'radial'}})
```

If `type` isn't specified it defaults to `linear`.

For a `linear` fill the angle of the gradient can also be specified (the default angle is 90 degrees):

```
worksheet.insert_textbox('B2', 'A textbox with angle gradient',
                         {'gradient': {'colors': ['#DDEBCF', '#9CB86E', '#156B13'],
                                       'angle': 45}})
```

Textbox Fonts

The following font properties can be set for the entire textbox:

```
name
size
bold
italic
underline
color
```

These properties correspond to the equivalent Worksheet cell Format object properties. See the The Format Class section for more details about Format properties and how to set them.

The font properties are:

- `name`: Set the font name:

```
{'font':  {'name': 'Arial'}}
```

- `size`: Set the font size:

```
{'font':  {'name': 'Arial', 'size': 9}}
```

- `bold`: Set the font bold property:

```
{'font':  {'bold': True}}
```

- `italic`: Set the font italic property:

```
{'font':  {'italic': True}}
```

- `underline`: Set the font underline property:

```
{'font':  {'underline': True}}
```

- `color`: Set the font color property. Can be a color index, a color name or HTML style RGB color:

```
{'font': {'color': 'red' }}
{'font': {'color': '#92D050'}}
```

Here is an example of Font formatting in a textbox:

```
worksheet.insert_textbox('B2', 'Some font formatting',
                         {'font': {'bold': True,
                                   'italic': True,
                                   'underline': True,
                                   'name': 'Arial',
                                   'color': 'red',
                                   'size': 14}})
```

Textbox Align

该`align`属性用于设置整个文本框的文本对齐方式：

```
worksheet.insert_textbox('B2', 'Alignment: middle - center',
                         {'align': {'vertical': 'middle',
                                    'horizontal': 'center'}})
```

可以在Excel中为文本框设置的对齐属性是：

```
{'align': {'vertical': 'top'}}      # Default
{'align': {'vertical': 'middle'}}
{'align': {'vertical': 'bottom'}}

{'align': {'horizontal': 'left'}}   # Default
{'align': {'horizontal': 'center'}}
```

默认的文本框对齐方式是：

```
worksheet.insert_textbox('B2', 'Default alignment',
                         {'align': {'vertical': 'top',
                                    'horizontal': 'left'}})

# Same as this:
worksheet.insert_textbox('B2', 'Default alignment')
```

sparklines detail

迷你图是Excel 2010+的一项功能，允许您向工作表单元格添加小图表。

add_sparkline（）方法

该 `add_sparkline()` 工作表方法用于迷你图添加到单元格或单元格范围：

```
worksheet.add_sparkline(0, 5, {'range': 'Sheet1!A1:E1'})
```

支持行列和A1样式表示法。有关 详细信息，请参阅使用单元格表示法。

`add_sparkline()` 必须在字典中传递参数。主要的迷你线参数是：

范围（必填）
类型
样式
标记
negative_points
轴
相反

其他不太常用的参数是：

地点
高点
低点
first_point
last_point
max
min
empty_cells
show_hidden
date_axis
weight
series_color
negative_color
markers_color
first_color
last_color
high_color
low_color

range

The `range` specifier is the only non-optional parameter.

It specifies the cell data range that the sparkline will plot:

```
worksheet.add_sparkline('F1', {'range': 'A1:E1'})
```

The `range` should be a 2D array. (For 3D arrays of cells see “Grouped Sparklines” below).

If `range` is not on the same worksheet you can specify its location using the usual Excel notation:

```
worksheet.add_sparkline('F1', {'range': 'Sheet2!A1:E1'})
```

If the worksheet contains spaces or special characters you should quote the worksheet name in the same way that Excel does:

```
worksheet.add_sparkline('F1', {'range': "'Monthly Data'!A1:E1"})
```

type

Specifies the type of sparkline. There are 3 available sparkline types:

```
line (default)
column
win_loss
```

For example:

```
worksheet.add_sparkline('F2', {'range': 'A2:E2',
                               'type': 'column'})
```

style

Excel provides 36 built-in Sparkline styles in 6 groups of 6. The `style` parameter can be used to replicate these and should be a corresponding number from 1 .. 36:

```
worksheet.add_sparkline('F2', {'range': 'A2:E2',
                               'type': 'column',
                               'style': 12})
```

The style number starts in the top left of the style grid and runs left to right. The default style is 1. It is possible to override color elements of the sparklines using the `_color` parameters below.

markers

Turn on the markers for `line` style sparklines:

```
worksheet.add_sparkline('A6', {'range': 'Sheet2!A1:J1',
                               'markers': True})
```

Markers aren't shown in Excel for `column` and `win_loss` sparklines.

negative_points

Highlight negative values in a sparkline range. This is usually required with `win_loss` sparklines:

```
worksheet.add_sparkline('A9', {'range': 'Sheet2!A1:J1',
                               'negative_points': True})
```

axis

Display a horizontal axis in the sparkline:

```
worksheet.add_sparkline('A10', {'range': 'Sheet2!A1:J1',
                                'axis': True})
```

reverse

Plot the data from right-to-left instead of the default left-to-right:

```
worksheet.add_sparkline('A24', {'range': 'Sheet2!A4:J4',
                                'type': 'column',
                                'style': 20,
                                'reverse': True})
```

weight

Adjust the default line weight (thickness) for `line` style sparklines:

```
worksheet.add_sparkline('F2', {'range': 'A2:E2',
                               'weight': 0.25})
```

The weight value should be one of the following values allowed by Excel:

```
0.25, 0.5, 0.75, 1, 1.25, 2.25, 3, 4.25, 6
```

high_point, low_point, first_point, last_point

Highlight points in a sparkline range:

```
worksheet.add_sparkline('A7', {'range': 'Sheet2!A1:J1',
                               'high_point': True,
                               'low_point': True,
                               'first_point': True})
```

max, min

Specify the maximum and minimum vertical axis values:

```
worksheet.add_sparkline('F1', {'range': 'A1:E1',
                               'max': 0.5,
                               'min': -0.5})
```

As a special case you can set the maximum and minimum to be for a group of sparklines rather than one:

```
'max': 'group'
```

See “Grouped Sparklines” below.

empty_cells

Define how empty cells are handled in a sparkline:

```
worksheet.add_sparkline('F1', {'range': 'A1:E1',
                               'empty_cells': 'zero'})
```

The available options are:

- `gaps`: show empty cells as gaps (the default).
- `zero`: plot empty cells as 0.
- `connect`: Connect points with a line (“line” type sparklines only).

show_hidden

Plot data in hidden rows and columns:

```
worksheet.add_sparkline('F3', {'range': 'A3:E3',
                               'show_hidden': True})
```

Note, this option is off by default.

date_axis

为迷你图指定替代日期轴。如果绘制的数据不是固定宽度间隔，这将非常有用：

```
worksheet.add_sparkline('F3', {'range': 'A3:E3',
                               'date_axis': 'A4:E4'})
```

日期范围中的单元格数应与数据范围中的单元格数相对应。

series_color

可以使用以下参数覆盖迷你图样式的颜色：

```
series_color
negative_color
markers_color
first_color
last_color
high_color
low_color
```

应将颜色指定为HTML样式`#rrggbb`十六进制值：

```
worksheet.add_sparkline('A18', {'range': 'Sheet2!A2:J2',
                                'type': 'column',
                                'series_color': '#E965E0'})
```

Grouped Sparklines

的`add_sparkline()`工作表的方法，可以多次使用作为在工作表中被要求写入尽可能多的迷你图。

但是，有时需要对连续的迷你图进行分组，以便将应用于一个的更改应用于所有。在Excel中，这是通过为数据选择3D范围的单元格以及为数据选择`range`2D范围的单元格来实现的`location`。

在XlsxWriter中，您可以通过将值的数组引用传递给`location`和来模拟这个`range`：

```
worksheet.add_sparkline('A27', {'location': ['A27',   'A28',   'A29'],
                                'range':    ['A5:J5', 'A6:J6', 'A7:J7']})
```