

- ParagraphFormat.line_spacing_rule : 选择一个 WD_LINE_SPACING (几倍行距);
- ParagraphFormat.line_spacing : 间距中的“设置值”长度 ;
- ParagraphFormat.widow_control : 孤行控制 , 用 True 表示设置 , None 表示继承 Style 设置 ;
- ParagraphFormat.keep_with_next : 与下段同页 , 用 True 表示设置 , None 表示继承 Style 设置 ;
- ParagraphFormat.keep_together : 段中不分页 , 用 True 表示设置 , None 表示继承 Style 设置 ;
- ParagraphFormat.page_break_before : 段前分页 , 用 True 表示设置 , None 表示继承 Style 设置。

例如通过下面的实例文件 python-docx14.py , 演示了设置段落递进的左对齐样式的过程。

源码路径 : 光盘:daima\7\7-2\python-docx14.py

```
doc = Document()
for i in range(10):
    p = doc.add_paragraph(u'段落 %d' % i)
    style = doc.styles.add_style('UserStyle%d' % i, WD_STYLE_TYPE.PARAGRAPH)
    style.paragraph_format.left_indent = Cm(i)
    p.style = style

doc.save('style-3.docx')
```

执行后将会创建一个包含指定段落样式文本的 Word 文件 style-3.docx , 如图 7-1 所示。

段落 0

段落 1

段落 2

段落 3

段落 4

段落 5

段落 6

段落 7

段落 8

段落 9

在开发 Python 应用程序的过程中，经常需要将一些数据处理并保存成不同的文件格式，例如 Office、PDF 和 CSV 等文件格式。在本章的内容中，将详细讲解在 Python 第三方库将数据处理成特殊文件格式的知识，为读者步入本书后面知识的学习打下基础。

7.1 使用 Tablib 模块

在 Python 程序中，可以使用第三方模块 Tablib 将数据导出为各种不同的格式，包括 Excel、JSON、HTML、Yaml、CSV 和 TSV 等格式。在使用之前需要先安装 Tablib，安装命令如下所示。

```
pip install tablib
```

在接下来的内容中，将详细讲解使用 Tablib 模块的知识。

7.1.1 基本用法

1. 创建 Dataset (数据集)

在 Tablib 模块中，使用 `tablib.Dataset` 创建一个简单的数据集对象实例：

```
data = tablib.Dataset()
```

接下来就可以填充数据集对和数据。

2. 添加 Rows (行)

假如我们想收集一个简单的人名列表，首先看下面的实现代码：

```
#名称的集合
names = ['Kenneth Reitz', 'Bessie Monke']

for name in names:
    #分割名称
    fname, lname = name.split()

    # 将名称添加到数据集
    data.append([fname, lname])
```

在 Python 中我们可以通过下面的代码获取人名：

```
>>> data.dict
[('Kenneth', 'Reitz'), ('Bessie', 'Monke')]
```

3. 添加 Headers (标题)

7.1.2 操作数据集中的指定行和列

在下面的实例文件 `Tablib01.py` 中，演示了使用 Tablib 模块操作操作数据集中的指定行和列的过程。

- ParagraphFormat.line_spacing_rule : 选择一个 WD_LINE_SPACING (几倍行距);
- ParagraphFormat.line_spacing : 间距中的“设置值”长度 ;
- ParagraphFormat.widow_control : 孤行控制 , 用 True 表示设置 , None 表示继承 Style 设置 ;
- ParagraphFormat.keep_with_next : 与下段同页 , 用 True 表示设置 , None 表示继承 Style 设置 ;
- ParagraphFormat.keep_together : 段中不分页 , 用 True 表示设置 , None 表示继承 Style 设置 ;
- ParagraphFormat.page_break_before : 段前分页 , 用 True 表示设置 , None 表示继承 Style 设置。

例如通过下面的实例文件 python-docx14.py , 演示了设置段落递进的左对齐样式的过程。

源码路径 : 光盘:daima\7\7-2\python-docx14.py

```
doc = Document()
for i in range(10):
    p = doc.add_paragraph(u'段落 %d' % i)
    style = doc.styles.add_style('UserStyle%d' % i, WD_STYLE_TYPE.PARAGRAPH)
    style.paragraph_format.left_indent = Cm(i)
    p.style = style

doc.save('style-3.docx')
```

执行后将会创建一个包含指定段落样式文本的 Word 文件 style-3.docx , 如图 7-1 所示。

段落 0

段落 1

段落 2

段落 3

段落 4

段落 5

段落 6

段落 7

段落 8

段落 9

图 7-1 文件 style-3.docx 的内容

(3) 样式管理器

在 Word 中自带了多种样式，我们可以使用库 python-docx 中的 Document.styles 集合来访问 builtin 属性为 True 的自带样式。当然，开发者通过 add_style() 函数增加的样式，也会被放在 styles 集合中。如果是开发者自己创建的样式，如果将其属性 hidden 和 quick_style 分别设置为 False 和 True，则可以将这个自建样式添加到 Word 快速样式管理器中。例如在下面的实例文件 python-docx15.py，演示了开发者自定义创建样式的过程。

源码路径：光盘\daima\7-2\python-docx15.py

```
doc = Document()
for i in range(10):
    p = doc.add_paragraph(u'段落 %d' % i)
    style = doc.styles.add_style('UserStyle%d' % i, WD_STYLE_TYPE.PARAGRAPH)
    style.paragraph_format.left_indent = Cm(i)
    p.style = style
    if i == 7:
        style.hidden = False
        style.quick_style = True

for style in doc.styles:
    print(style.name, style.builtin)

doc.paragraphs[3].style = doc.styles['Subtitle']
doc.save('style-4.docx')
```

通过上述代码，在 Word 文件 style-4.docx 中自定义创建了 9 种 (UserStyle1~ UserStyle9) 样式。如图 7-1 所示。



图 7-1 文件 style-4.docx 中的样式和内容

7.2.4 使用 xlrd 和 xlwt 读写 Excel

<https://blog.csdn.net/Tulaimes/article/details/71172778>

1. 库 xlrd

在 Python 程序中，可以使用库 xlrd 读取 Excel 文件的内容。安装库 xlrd 的命令如下所示：

```
pip install xlrd
```

使用库 xlrd 读取 Excel 文件的步骤如下：

(1) 通过如下代码导入模块：

```
import xlrd
```

(2) 读取 Excel 数据，例如下面代码的意思是打开 unit 表，将表中数据读进 data 中。

```
xlrd.open_workbook(excel路径)
data = xlrd.open_workbook('unit.xlsx')l
```

(3) 获取一个工作表 (Sheet)

获取工作表的方法有两种：通过索引顺序获取和通过工作表名字获取，下面是这两种获取方

式的演示代码：

```
sheet = data.sheets()[0] #通过索引顺序获取第一个工作表
sheet = data.sheet_by_index(0) #通过索引顺序获取第一个工作表
sheet = data.sheet_by_name(u'Sheet1') #通过名称获取l
```

在上述代码中，sheet.name 表示工作表的名字。

(4) 获取每一行或每一列的信息，例如通过如下代码获取一个工作表的总行数和总列数。

```
nrows = sheet.nrows #行数
ncols = sheet.ncols #列数l
```

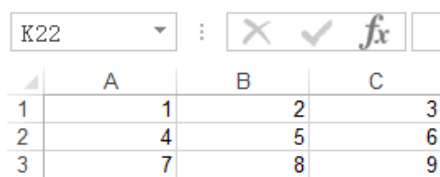
例如通过如下代码获取一个工作表的整行或整列的值 (数组)：

```
sheet.row_values(n) #获取第n行整行的值，返回一个数组
sheet.col_values(m) #获取第m行整行的值，返回一个数组l
```

例如通过如下代码获取一个单元格的值：

```
sheet.cell(i, j).value #i行索引, j列索引
```

假设存在一个 Excel 文件 example.xlsx，其内容如图 7-1 所示。



	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9

图 7-1 文件 example.xlsx 的内容

例如在下面的实例文件 ex01.py 中，演示了使用库 xlrd 读取指定 Excel 文件内容的过程。

源码路径：光盘\daima\7\7-2\ex01.py

```
import xlrd
#打开excel
data = xlrd.open_workbook('example.xlsx')
#查看文件中包含sheet的名称
data.sheet_names()
#得到第一个工作表，或者通过索引顺序或工作表名称
table = data.sheets()[0]
```

```

table = data.sheet_by_index(0)
table = data.sheet_by_name(u'Sheet1')
#获取行数和列数
nrows = table.nrows
ncols = table.ncols
print(nrows)
print(ncols)
#循环行,得到索引的列表
for rownum in range(table.nrows):
    print(table.row_values(rownum))
#分别使用行列索引
cell_A1 = table.row(0)[0].value
cell_A2 = table.col(1)[0].value
print(cell_A1)
print(cell_A2)

```

执行后会输出：

```

3
3
[1.0, 2.0, 3.0]
[4.0, 5.0, 6.0]
[7.0, 8.0, 9.0]
1.0
6.0
1.0

```

2. 库 xlwt

在 Python 程序中,可以使用库 xlrd 向 Excel 文件中写入内容。安装库 xlwt 的命令如下所示：

```
pip install xlwt
```

使用库 xlw 写入 Excel 文件的步骤如下：

(1) 通过如下代码导入 xlwt：

```
import xlwt
```

(2) 通过如下代码创建 workbook，其实就是 Excel：

```
workbook = xlwt.Workbook(encoding = 'ascii')
```

(3) 通过如下代码创建表：

```
worksheet = workbook.add_sheet('My Worksheet')
```

(4) 通过如下代码向单元格中写入内容：

```
worksheet.write(0, 0, label = 'Row 0, Column 0 Value')
```

(5) 保存 Excel 单元格，例如下面的的代码：

```
workbook.save('Excel_Workbook.xls')
```

例如在下面的实例文件 ex02.py 中，演示了使用库 xlwt 将指定内容写入到 Excel 文件并创建

Excel 文件的过程。

源码路径：光盘:daima\7\7-2\ex02.py

```

import xlwt
from datetime import datetime

style0 = xlwt.easyxf('font: name Times New Roman, color-index red, bold on',
    num_format_str='#,##0.00')
style1 = xlwt.easyxf(num_format_str='D-MMM-YY')#当前日期

wb = xlwt.Workbook()
ws = wb.add_sheet('A Test Sheet')          #sheet的名字

ws.write(0, 0, 1234.56, style0)            #第1个cell的内容

```

```

ws.write(1, 0, datetime.now(), style1)          #第2个cell的内容
ws.write(2, 0, 1)                               #第3个cell的内容
ws.write(2, 1, 1)                               #第4个cell的内容
ws.write(2, 2, xlwt.Formula("A3+B3"))          #第5个cell的内容

wb.save('example02.xls')

```

执行后会将指定内容写入到文件 example02.xls 中，如图 7-1 所示。

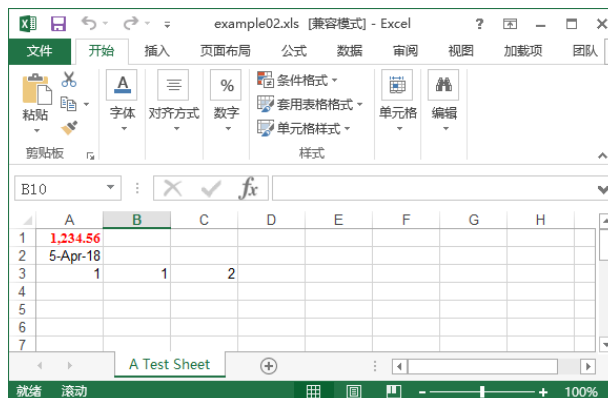


图 7-1 文件 example02.xls 中的内容

7.2.5 使用 xlswriter

<https://github.com/jmcnamara/XlsxWriter/tree/master/examples>

<http://ju.outofmemory.cn/entry/106037>

<https://www.jianshu.com/p/7d6f53e3e6e9>

在 Python 程序中，可以使用库 xlswriter 操作 Excel 文件。安装命令如下所示：

```
pip install xlswriter
```

使用库 xlswriter 的基本流程如下所示：

(1) 首先创建一个 Excel 的文档：

```
workbook = xlswriter.Workbook(dir)
```

(2) 在文档中创建表

```

table_name = 'sheet1'
worksheet = workbook.add_worksheet(table_name) # 创建一个表名为 'sheet1' 的表，并返回这个表对象

```

(3) 创建表后，就可以在表格上面进行写入操作：

```
worksheet.write_column('A1', 5) # 在A1单元格写入数字5
```

我们可以想改输入内容的格式，例如设置字体颜色加粗、斜体和日期格式等，这时可以通过

使用 xlswriter 提供的格式类实现。通过下面的代码写入了一个红色粗体的日期类。

```

import datetime
# 需要先把字符串格式化日期
date_time = datetime.datetime.strptime('2017-1-25', '%Y-%m-%d')
# 定义一个格式类，粗体的红色的日期
date_format = workbook.add_format({'bold': True, 'font_color': 'red', 'num_format': 'yyyy-mm-dd'})
# 写入该格式类
worksheet.write_column('A2', date_time, date_format)

```

1. 创建一个表格

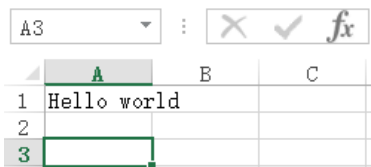
例如在下面的实例文件 `xlsxwriter01.py` 中 ,演示了使用库 `xlsxwriter` 创建一个指定内容 Excel 文件的过程。

```
源码路径：光盘\daima\7-2\xlsxwriter01.py
import xlsxwriter # 导入模板

workbook = xlsxwriter.Workbook('hello.xlsx') # 创建一个名为 hello.xlsx 赋值给workbook
worksheet = workbook.add_worksheet() # 创建一个默认工作簿 赋值给worksheet
# 工作簿也支持命名，
# 如：workbook.add_worksheet('hello')

worksheet.write('A1', 'Hello world') # 使用工作簿在 A1地方 写入Hello world
workbook.close() # 关闭工作簿
```

执行后会创建一个 Excel 文件 `hello.xlsx` , 如图 7-1 所示。



	A	B	C
1	Hello world		
2			
3			

图 7-1 文件 `hello.xlsx` 的内容

在下面的实例文件 `xlsxwriter02.py` 中 ,演示了使用库 `xlsxwriter` 定 Excel 文件中批量写入指定内容的过程。

```
源码路径：光盘\daima\7-2\xlsxwriter02.py
import xlsxwriter

workbook = xlsxwriter.Workbook('Expenses01.xlsx')
worksheet = workbook.add_worksheet()

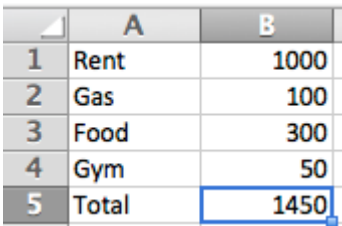
# 需要写入的数据
expenses = ([ 'Rent', 1000],
            [ 'Gas', 100],
            [ 'Food', 300],
            [ 'Gym', 50],
            )

# 行跟列的初始位置
row = 0
col = 0

# .write方法 write (行,列,写入的内容,样式)
for item, cost in (expenses):
    worksheet.write(row, col, item) # 在第一列的地方写入item
    worksheet.write(row, col + 1, cost) # 在第二列的地方写入cost
    row + 1 # 每次循环行数发生改变

worksheet.write(row, 0, 'Total')
worksheet.write(row, 1, '=SUM(B1:B4)') # 写入公式
```

执行后会创建一个 Excel 文件 `Expenses01.xlsx` , 如图 7-1 所示。



	A	B
1	Rent	1000
2	Gas	100
3	Food	300
4	Gym	50
5	Total	1450

图 7-1 文件 Expenses02.xlsx 的内容

2. 设置表格样式

表格样式包含字体、颜色、模式、边框和数字格式等，在设置表格样式需要使用函数

add_format()，库 xlsxwriter 中包含的样式信息如表 7-1 所示。

表 7-1 库 xlsxwriter 中包含的样式信息

类别	描述	属性	方法名
字体	字体	font_name	set_font_name()
	字体大小	font_size	set_font_size()
	字体颜色	font_color	set_font_color()
	加粗	bold	set_bold()
	斜体	italic	set_italic()
	下划线	underline	set_underline()
	删除线	font_strikeout	set_font_strikeout()
	上标/下标	font_script	set_font_script()
数字	数字格式	num_format	set_num_format()
保护	表格锁定	locked	set_locked()
	隐藏公式	hidden	set_hidden()
对齐	水平对齐	align	set_align()
	垂直对齐	valign	set_align()
	旋转	rotation	set_rotation()
	文本包装	text_wrap	set_text_wrap()
	底端对齐	text_justlast	set_text_justlast()
	中心对齐	center_across	set_center_across
	缩进	indent	set_indent()
	缩小填充	shrink	set_shrink()
模式	表格模式	pattern	set_pattern()
	背景颜色	bg_color	set_bg_color()
	前景颜色	fg_color	set_fg_color()
边框	表格边框	border	set_border()
	底部边框	bottom	set_bottom()
	上边框	top	set_top()
	右边框	right	set_right()
	边框颜色	border_color	set_border_color()
	底部颜色	bottom_color	set_bottom_color()

	顶部颜色	top_color	set_top_color()
	左边颜色	left_color	set_left_color()
	右边颜色	right_color	set_right_color()

在下面的实例文件 `xlsxwriter03.py` 中，演示了使用库 `xlsxwriter` 创建指定 Excel 格式内容的过程。

源码路径：光盘:daima\7\7-2\xlsxwriter03.py

```
# 建文件及sheet.
workbook = xlsxwriter.Workbook('Expenses03.xlsx')
worksheet = workbook.add_worksheet()
# 设置粗体，默认是False
bold = workbook.add_format({'bold': True})
# 定义数字格式
money = workbook.add_format({'num_format': '$#,##0'})
#带自定义粗体bold格式写表头
worksheet.write('A1', 'Item', bold)
worksheet.write('B1', 'Cost', bold)
#写入表中的数据.
expenses = (
    ['Rent', 1000],
    ['Gas', 100],
    ['Food', 300],
    ['Gym', 50],
)

#从标题下面的第一个单元格开始 .
row = 1
col = 0

# 迭代数据并逐行地写出它
for item, cost in expenses:
    worksheet.write(row, col, item) # 带默认格式写入
    worksheet.write(row, col + 1, cost, money) # 带自定义money格式写入
    row += 1

# 用公式计算总数 .
worksheet.write(row, 0, 'Total', bold)
worksheet.write(row, 1, '=SUM(B2:B5)', money)

workbook.close()
```

执行后会创建一个 Excel 文件 `Expenses03.xlsx`，表格中的字体样式是我们自己定义的。如图 7-1 所示。

	A	B	C
1	Item	Cost	
2	Rent	\$1,000	
3	Gas	\$100	
4	Food	\$300	
5	Gym	\$50	
6	Total	\$1,450	
7			

图 7-1 文件 `Expenses03.xlsx` 的内容

3. 插入图像

在下面的实例文件 `xlsxwriter04.py` 中，演示了使用库 `xlsxwriter` 向指定 Excel 文件中插入指定图像的过程。

源码路径：光盘:daima\7\7-2\xlsxwriter04.py

```
# 创建一个新Excel文件并添加工作表
workbook = xlsxwriter.Workbook('demo.xlsx')
worksheet = workbook.add_worksheet()

# 展开第一栏，使正文更清楚
worksheet.set_column('A:A', 20)

# 添加一个粗体格式用于高亮单元格
bold = workbook.add_format({'bold': True})

# 写一些简单的文字。
worksheet.write('A1', 'Hello')

# 设置文本与格式
worksheet.write('A2', 'World', bold)

# 写一些数字，行/列符号
worksheet.write(2, 0, 123)
worksheet.write(3, 0, 123.456)
#插入图像
worksheet.insert_image('B5', '123.png')
workbook.close()
```

执行后会创建一个包含指定图像内容的 Excel 文件 demo.xlsx，如图 7-1 所示。

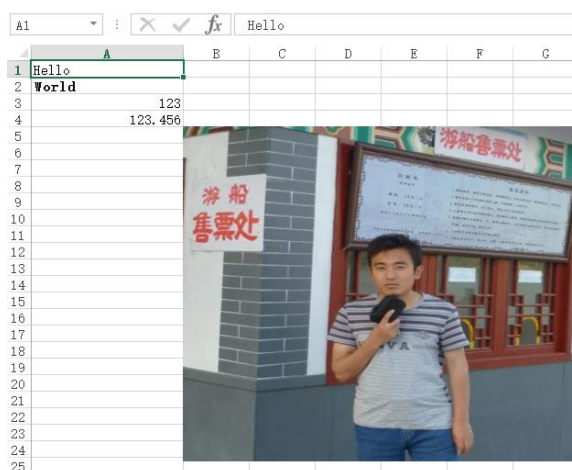


图 7-1 文件 demo.xlsx 的内容

4. 插入图表

Excel 的核心功能之一便是将表格内的数据生成统计图表，是整个数据变得更加直观。通过使用库 xlsxwriter，可以将 Excel 表格内的数据生成图表。Excel 支持两种类型的图表，其中第一种类型分别有如下所示的 9 大类：

- area：面积图；
- bar：转置直方图；
- column：柱状图；
- line：直线图；

- pie : 饼状图 ;
- doughnut : 环形图 ;
- scatter : 散点图 ;
- stock : 股票趋势图 ;

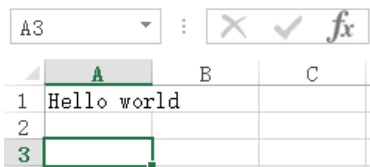
例如在下面的实例文件 `xlsxwriter01.py` 中 ,演示了使用库 `xlsxwriter` 创建一个指定内容 Excel 文件的过程。

```
源码路径：光盘\daima\7-2\xlsxwriter01.py
import xlsxwriter # 导入模板

workbook = xlsxwriter.Workbook('hello.xlsx') # 创建一个名为 hello.xlsx 赋值给workbook
worksheet = workbook.add_worksheet() # 创建一个默认工作簿 赋值给worksheet
# 工作簿也支持命名，
# 如：workbook.add_worksheet('hello')

worksheet.write('A1', 'Hello world') # 使用工作簿在 A1地方 写入Hello world
workbook.close() # 关闭工作簿
```

执行后会创建一个 Excel 文件 `hello.xlsx`，如图 7-1 所示。



	A	B	C
1	Hello world		
2			
3			

图 7-1 文件 `hello.xlsx` 的内容

在下面的实例文件 `xlsxwriter02.py` 中，演示了使用库 `xlsxwriter` 定 Excel 文件中批量写入指定内容的过程。

```
源码路径：光盘\daima\7-2\xlsxwriter02.py
import xlsxwriter

workbook = xlsxwriter.Workbook('Expenses01.xlsx')
worksheet = workbook.add_worksheet()

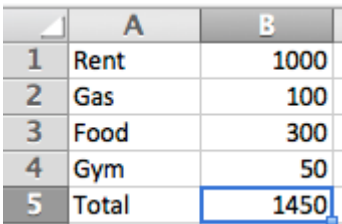
# 需要写入的数据
expenses = ([ 'Rent', 1000],
            [ 'Gas', 100],
            [ 'Food', 300],
            [ 'Gym', 50],
            )

# 行跟列的初始位置
row = 0
col = 0

# .write方法 write（行,列,写入的内容,样式）
for item, cost in (expenses):
    worksheet.write(row, col, item) # 在第一列的地方写入item
    worksheet.write(row, col + 1, cost) # 在第二列的地方写入cost
    row + 1 # 每次循环行数发生改变

worksheet.write(row, 0, 'Total')
worksheet.write(row, 1, '=SUM(B1:B4)') # 写入公式
```

执行后会创建一个 Excel 文件 `Expenses01.xlsx`，如图 7-1 所示。



	A	B
1	Rent	1000
2	Gas	100
3	Food	300
4	Gym	50
5	Total	1450