

安装

2020年6月11日 10:03

1、找到pip3.exe所在的文件夹，复制路径

我的路径是： C:\Users\孙艺航\AppData\Local\Programs\Python
\Python37\Scripts

2、按Win+R,输入CMD确定

3、进入后，先输入cd 路径 回车

4、输入 pip3 install python-docx 回车

常用代码

2020年6月21日 9:27

1.导入模块

2020年6月21日 9:27

```
from docx import Document
from docx.enum.text import WD_ALIGN_PARAGRAPH #设置对象居中、对齐等
from docx.enum.text import WD_TAB_ALIGNMENT,WD_TAB_LEADER #设置制表符等
from docx.shared import Inches #设置图像大小
from docx.shared import Pt #设置像素、缩进等
from docx.shared import RGBColor #设置字体颜色
from docx.shared import Length #设置宽度
```

2.字号

2020年6月21日

9:28

字号	磅值
八号	5
七号	5.5
小六	6.5
六号	7.5
小五	9
五号	10.5
小四	12
四号	14
小三	15
三号	16
小二	18
二号	22
小一	24
一号	26
小初	36
初号	42

01.初识Python-docx库

2020年6月19日 22:29

一、文件.paragraphs 得到的是一个列表，包含了每个段落的实例，可以索引、切片、遍历

from 从 docx这个文件中，导入一个叫Document的一个东西，Document是文档的意思

```
from docx import Document
```

文件 = Document() 可以理解为 Document就是一个类，这个操作也就是实例化的过程，生成对象为：文件

```
文件 = Document('c:/练习.docx')
```

文件.paragraphs # 返回文档中每个段落集合，是一个列表，可以通过索引获取

```
print(文件.paragraphs)
```

```
print(文件.paragraphs[0])
```

```
print(文件.paragraphs[0:2])
```

二、段落.text 得到该段落的文字内容

```
from docx import Document
```

```
文件 = Document('c:/练习.docx')
```

```
for 段落 in 文件.paragraphs:
```

```
    print(段落.text)
```

三、块与文字

段落.runs 得到一个列表，包含了每个文字块，可索引、切片、遍历

文字.text 得到该文字块的文字内容

```
from docx import Document
```

```
文件 = Document('c:/练习.docx')
```

```
段落 = 文件.paragraphs[0]
```

```
块 = 段落.runs
```

```
for 文字 in 块:
```

```
    print(文字.text)
```

```
from docx import Document
```

```
文件 = Document('c:/练习2.docx')
```

```
段落 = 文件.paragraphs[0]
```

```
块 = 段落.runs
```

```
for 文字 in 块:
```

```
    print(文字.text)
```

练习：在练习2.docx中搜索 ‘孙兴华’ 出现的次数

```
from docx import Document
```

```
文件 = Document('c:/练习2.docx')
```

```
计数 = 0
```

```
for 段落 in 文件.paragraphs:
```

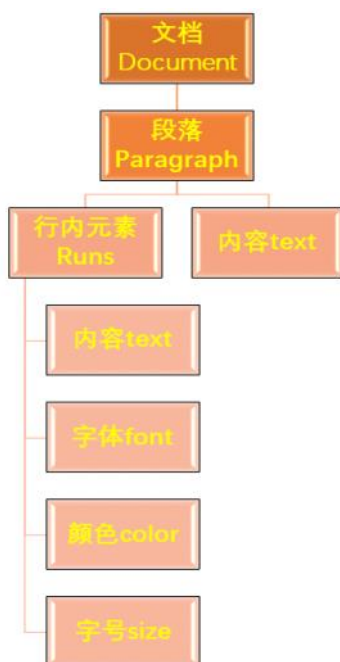
```
    if '孙兴华' in 段落.text:
```

```
        计数 += 1
```

```
print(计数)
```

1.1.文档的基本结构

2020年6月19日 23:56



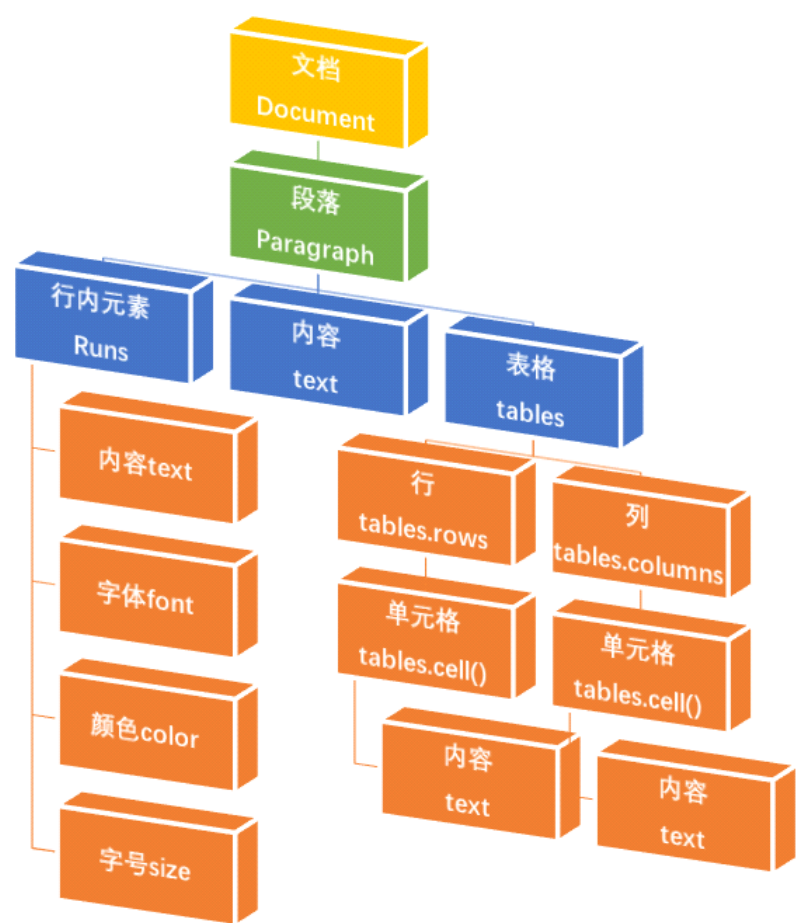
python-docx将整个文章看做是一个Document对象，其基本结构如下：

每个Document包含许多个代表“段落”的Paragraph对象，存放在document.paragraphs中。

每个Paragraph都有许多个代表“行内元素”的Run对象，存放在paragraph.runs中。

1.2.文件的完整结构

2020年6月20日 0:00



1.3.读取docx文件中表格数据

2020年6月20日 2:51

例：查询练习2中“孙兴华”三个字出现的次数

```
from docx import Document
# 打开文档
文件 = Document('c:/练习2.docx')
i=0
j=0
# 遍历所有段落文本
for 段落 in 文件.paragraphs:
    if '孙兴华' in 段落.text:
        i+=1
# 遍历所有单元格文本
for 表 in 文件.tables:
    for 行 in 表.rows:
        for 单元格 in 行.cells:
            if '孙兴华' in 单元格.text:
                j+=1
print(f'孙兴华在文档中一共出现了{i+j}次')
```


05.读取Word文字

2020年6月20日 19:30

一、读取Word所有内容

```
from docx import Document
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    print(段落.text)
```

二、读取一级标题

```
from docx import Document
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if 段落.style.name == 'Heading 1':
        print(段落.text)
```

三、读取二级标题

```
from docx import Document
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if 段落.style.name == 'Heading 2':
        print(段落.text)
```

四、读取所有标题【使用正则 笔记2.4】

```
from docx import Document
import re
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if re.match("^Heading \d+$",段落.style.name):
        print(段落.text)
```

五、读取正文

```
from docx import Document
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if 段落.style.name == 'Normal':
        print(段落.text)
```

六、读取标题名称

```
from docx.enum.style import WD_STYLE_TYPE
from docx import Document
文件 = Document('c:/练习3.docx')
标题 = 文件.styles
for i in 标题:
    if i.type == WD_STYLE_TYPE.PARAGRAPH:
        print(i.name)
```

06.写入Word文字

2020年6月20日 21:12

一、添加标题

```
from docx import Document
文件 = Document('c:/练习3.docx')
文件.add_heading("我是一级标题",level=1)
文件.save('c:/练习4.docx')
```

二、添加正文

```
from docx import Document
文件 = Document('c:/练习3.docx')
文件.add_paragraph("我是正文1234567")
文件.save('c:/练习4.docx')
```

三、添加分页符

```
from docx import Document
文件 = Document('c:/练习3.docx')
文件.add_page_break()
文件.save('c:/练习4.docx')
```

四、添加文字块

```
from docx import Document
文件 = Document('c:/练习3.docx')
a = 文件.add_paragraph('我是正文在我后面添加的文字会被设置格式：')
a.add_run('加粗').bold = True
a.add_run('普通')
a.add_run('斜体').italic = True
文件.save('c:/练习4.docx')
```


6.1 段落的定位

2020年6月21日 8:32

```
from docx import Document
```

```
文件 = Document('c:/练习9.docx')
```

```
print(len(文件.paragraphs))
```

```
段落 = 文件.paragraphs[1]
```

```
print(段落.text)
```

6.2 指定段落处添加段落

2020年6月21日 8:32

通过选择段落，获取段落对象，可以使用insert_paragraph_before()函数进行设置，其参数同add_paragraph()

```
from docx import Document
```

```
文件 = Document('c:/练习9.docx')
```

```
段落 = 文件.paragraphs[1] # 获取第二个段落
```

```
段落.insert_paragraph_before('这是添加的新的第二个段落') # 在第二个段落处插入
```

```
文件.save('c:/5.docx')
```

07.插入图片

2020年6月20日 21:33

添加图片 文件.add_picture(图片地址, width = 宽度, height = 高度)

一、在word文档中插入图片，并设置大小

```
from docx.shared import Cm
from docx import Document
文件 = Document('c:/练习4.docx')
# 文件.add_picture('c:/赵丽颖.jpg')
文件.add_picture('c:/赵丽颖.jpg', width = Cm(13), height = Cm(8)) # Cm是厘米
文件.save('c:/练习5.docx')
```

二、在指定表的单元格中插入图片

```
from docx.shared import Cm
from docx import Document
文件 = Document('c:/练习4.docx')
run = 文件.tables[0].cell(0,0).paragraphs[0].add_run()
# run.add_picture('c:/赵丽颖.jpg')
run.add_picture('c:/赵丽颖.jpg', width = Cm(5), height = Cm(3)) # Cm是厘米
文件.save('c:/练习5.docx')
```

只给一个尺寸，另一个尺寸会自动计算

7.1 按照比例设置图片

2020年6月21日 10:36

Shape 对象代表文档中的图形对象，InlineShape 代表文档中的嵌入式图形对象。
所谓嵌入式图形对象，是指将图像作为文字处理，在排版上以文字的方式进行排版

一、以李小龙为基准将赵丽颖缩小

```
from docx import Document
```

```
文件 = Document('c:/练习4.docx')
```

```
图片1 = 文件.paragraphs[1].add_run().add_picture('c:/赵丽颖.jpg') # 在文档第2个段落里添加图片
```

```
图片2 = 文件.paragraphs[2].add_run().add_picture('c:/李小龙.jpg') # 在文档第3个段落里添加图片
```

```
print('原始图像和当前图像的高度', 文件.inline_shapes[0].height,文件.inline_shapes[1].height) # 打印原始图片大小
```

```
print('当前图像和原始图像的高度比值', 文件.inline_shapes[1].height / 文件.inline_shapes[0].height) # 打印当前图片高度比例
```

```
print('当前图像和原始图像的宽度比值', 文件.inline_shapes[1].width / 文件.inline_shapes[0].width) # 打印当前图片宽度比例
```

```
图片1.height = int(文件.inline_shapes[0].height * 0.707808564231738) # 按照比例设置图片高度
```

```
图片1.width = int(文件.inline_shapes[0].width * 0.7861635220125787) # 按照比例设置图片宽度
```

```
文件.save('c:/练习5.docx')
```

二、以赵丽颖为基准将李小龙放大

```
from docx import Document
```

```
文件 = Document('c:/练习4.docx')
```

```
图片1 = 文件.paragraphs[1].add_run().add_picture('c:/赵丽颖.jpg') # 在文档第2个段落里添加图片
```

```
图片2 = 文件.paragraphs[2].add_run().add_picture('c:/李小龙.jpg') # 在文档第3个段落里添加图片
```

```
print('原始图像和当前图像的高度', 文件.inline_shapes[0].height,文件.inline_shapes[1].height) # 打印原始图片大小
```

```
print('当前图像和原始图像的高度比值', 文件.inline_shapes[0].height / 文件.inline_shapes[1].height) # 打印当前图片高度比例
```

```
print('当前图像和原始图像的宽度比值', 文件.inline_shapes[0].width / 文件.inline_shapes[1].width) # 打印当前图片宽度比例
```

```
图片2.height = int(文件.inline_shapes[0].height * 1.4128113879003559) # 按照比例设置图片高度
```

```
图片2.width = int(文件.inline_shapes[0].width * 1.272) # 按照比例设置图片宽度
```

```
文件.save('c:/练习5.docx')
```

三、按照固定比例设置图像大小

```
from docx import Document
```

```
文件 = Document('c:/练习4.docx')
```

```
图片 = 文件.paragraphs[1].add_run().add_picture('c:/赵丽颖.jpg') # 在文档第2个段落里添加图片
```

```
图片.height = int(文件.inline_shapes[0].height * 0.80) # 按照50%比例设置图片高度
```

```
图片.width = int(文件.inline_shapes[0].width * 0.80) # 按照50%比例设置图片宽度
```

```
文件.save('c:/练习5.docx')
```

四、按照锁定比例设置图像大小

固定比例的另外一种常用的方式就是“锁定比例”。所谓锁定比例就是当设置高度和宽度中的任何一个时，另外一个也会按照相同的比例进行设置。一般经过读取原始图片的高度或者宽度，设置新的值，计算比例，最后将比例设置宽度或者高度。

```
from docx.shared import Cm
```

```
from docx import Document
```

```
文件 = Document('c:/练习4.docx')
```

```
图片 = 文件.paragraphs[1].add_run().add_picture('c:/赵丽颖.jpg') # 在文档第2个段落里添加图片
```

```
高度 = 图片.height # 读取图片原始大小高度
```

```
图片.height = Cm(5) # 设置图片高度为5cm
```

```
比例 = 图片.height / 高度 # 计算图片比例，新设定的高度除以原始高度
```

```
图片.width = int(文件.inline_shapes[0].width * 比例) # 按照比例设置图片宽度
```

```
文件.save('c:/练习5.docx')
```


7.2 设置图片的对齐方式

2020年6月21日 19:37

在插入图片时，经常使用run.add_picture()方法，本质上仍然是在段落中添加，所以，改变段落对齐方式，效果也作用到图片上，这个时候的图片是单独一个段落。

```
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT # 导入段落对齐包
from docx import Document
文件 = Document('c:/练习4.docx')
图片 = 文件.paragraphs[1].add_run().add_picture('c:/赵丽颖.jpg') # 在文档第2个段落里添加图片
文件.paragraphs[1].alignment = WD_PARAGRAPH_ALIGNMENT.CENTER # 设置文档中的第2个段落居中
文件.save('c:/练习5.docx')
```

附：对齐方式：

在WD_PARAGRAPH_ALIGNMENT可以实现LEFT、RIGHT、CENTER、JUSTY和DISTRIBUTE等5种对齐方式

WD_PARAGRAPH_ALIGNMENT.LEFT # 左对齐

WD_PARAGRAPH_ALIGNMENT.CENTER # 居中对齐

WD_PARAGRAPH_ALIGNMENT.RIGHT # 右对齐

WD_PARAGRAPH_ALIGNMENT.JUSTIFY # 两端对齐

WD_PARAGRAPH_ALIGNMENT.DISTRIBUTE # 分散对齐

7.3 图片的定位与删除

2020年6月21日 19:44

图像是通过run对象的add_picture()来添加的，而run对象是段落的一部分，所以通过删除段落可以删除图像。

```
from docx import Document
文件 = Document('c:/练习5.docx')
段落 = 文件.paragraphs[1] # 获取文档中的第2个段落对象
print('删除前图形图像的数量: ', len(文件.inline_shapes)) # 删除前图片的数量
段落.clear() # 删除段落
print('删除后图形图像的数量: ', len(文件.inline_shapes)) # 删除后图片的数量
文件.save('c:/11.docx')
```

08.表格

2020年6月20日 21:33

添加表格 文件.add_table(rows = 多少行, cols = 多少列)

```
from docx import Document
```

```
文件 = Document('c:/练习3.docx')
```

```
列表 = [
```

```
    ['姓名','性别','年龄'],
```

```
    ['孙兴华','男',20],
```

```
    ['赵丽颖','女',33],
```

```
    ['李小龙','男',80],
```

```
    ['叶问','男',127]
```

```
]
```

```
r = 5
```

```
c = 3
```

```
表 = 文件.add_table(rows = r, cols = c)
```

```
for 行号 in range(r):
```

```
    # print(行号)
```

```
    单元格 = 表.rows[行号].cells
```

```
    for 列号 in range(c):
```

```
        # print(列号)
```

```
        单元格[列号].text = str(列表[行号][列号])
```

```
文件.save('c:/练习5.docx')
```

8.1 添加表格行和列及查看行列数

2020年6月21日 2:54

```
from docx import Document
from docx.shared import Cm
文件 = Document('c:/练习2.docx')
# 文件.tables[0].add_row() # 表格最下方增加一行
文件.tables[0].add_column(Cm(5)) # 表格最右侧增加一列,一定要写列宽
print(f'现在第1个表的行是{len(文件.tables[0].rows)}, 列是{len(文件.tables[0].columns)}')
文件.save('c:/1.docx')
```

8.2 表格行和列的定位

2020年6月21日 3:03

在python-docx中表格中行或者列的定位主要通过

`table.rows`和`table.columns`两个属性获取行和列的总对象，然后使用索引获取指定的行或者列对象。获取表格中的第2行和第2列代码如下：

```
row = table.rows[1]
```

```
column = table.columns[1]
```

8.3 删除表格中的行、列

2020年6月21日 3:11

在表格中虽然单元格可以从column中的cells中来遍历，但是单元格是按行存储的，这点将在删除列的部分重点说明。

一、删除表格中的行

```
from docx import Document
```

```
文件 = Document('c:/练习2.docx')
print(f'此文件中共{len(文件.tables)}个表格')
表1 = 文件.tables[0] # 获取第一个表格
print(f'表1的共{len(表1.rows)}行, {len(表1.columns)}列')
行 = 表1.rows[1] # 获取表格中的第2行
行._element.getparent().remove(行._element) # 删除行
print(f'删除表格中的第2行后, 表1共{len(表1.rows)}行, {len(表1.columns)}列')
文件.save('c:/1.docx')
```

二、删除表的中列

列的删除则不能像删除行那样使用对应的remove()函数，因为在_Column中没有定义_element，但可以采用单元格进行删除。

```
from docx import Document
文件 = Document('c:/练习2.docx')
print(f'此文件中共{len(文件.tables)}个表格')
表1 = 文件.tables[0] # 获取第一个表格
print(f'表1的共{len(表1.rows)}行, {len(表1.columns)}列')
列 = 表1.table.columns[1] # 获取表1中第2列
for 单元格 in 列.cells: # 遍历列中的单元格
    单元格._element.getparent().remove(单元格._element) # 删除第2列的单元格
print(f'删除表格中的第2行后, 表1共{len(表1.rows)}行, {len(表1.columns)}列')
文件.save('c:/1.docx')
```

```
此文件中共1个表格
表1的共3行, 3列
删除表格中的第2行后, 表1共3行, 3列
```

通过cell的remove()方法可以删除表格的列，但是由于表格中的cell是按行存储，每行存储的cell的数量并没有变化，所以当删除单元格后，后续的单元格会补上。那么此时表格的存储形式将不是WORD文档表现出的效果，读取表1验证一下：

```
from docx import Document
文件 = Document('c:/1.docx')
表1 = 文件.tables[0] # 获取第一个表格
i = 0 # 标识行的序号
for 行 in 表1.rows: # 遍历取表格中的行
    i += 1 # 行的序号从1开始
    for 单元格 in 行.cells: # 遍历行中的单元格
        print(f'第{i}行: ', 单元格.text)
```

```
第1行: 孙兴华
第1行: 孙悟空
第1行: 赵丽颖
第2行: 唐僧
第2行: 孙兴华
第2行: 猪八戒
```

表格仍然按照每行3个元素存储，表格仍然是3行3列，但第3行是空行

8.4 表格的删除

2020年6月21日 8:19

```
from docx import Document
```

```
文件 = Document('c:/练习8.docx')
```

```
print(f'文件中表格总数:{len(文件.tables)}个')
```

```
表 = 文件.tables[1] # 获取第二个表格
```

```
表._element.getparent().remove(表._element) # 删除表格
```

```
print(f'删除后的表格总数:{len(文件.tables)}个')
```

```
文件.save('c:/5.docx')
```

8.5 设置单元格的值

2020年6月21日 19:49

表格中单元格的值有两种赋值方式，一种是直接为cell.text属性赋值来实现，另外一种是通过获取或者添加单元格中的段落，然后使用段落中的text属性赋值实现

```
from docx import Document
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT # 导入段落对齐方式
文件 = Document('c:/练习.docx')
# 方法1
表 = 文件.add_table(3, 3) # 为文档新增3行3列的表格
表.cell(0, 0).text = '孙兴华' # 为表格的 (0, 0) 位置单元格赋值
# 方法2
段落 = 表.cell(0, 1).paragraphs[0] # 获取表格 (0, 1) 位置单元格中的段落
段落.text = '赵丽颖'
段落 = 表.cell(0, 1).add_paragraph('第二个段落居中')
段落.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER # 设置段落居中
文件.save('c:/11.docx')
```

注：第一种赋值方式，只能更改单元格的值，无法设置单元格中数据的样式，并且整个单元格只能是一个段落；而第二种赋值方式，使用了段落，在单元格赋值的基础上还能增加新的段落，并设置段落的样式和字体的样式

8.6 表格的录入

2020年6月21日 19:49

在表格中使用`table.cell(col_index, row_index)`来定位单元格，使用`cell.text`属性设置单元格的值。新建表格并将第一行设置为表头，从第二行开始作为数据的开始行

一、简单直接的方式进入录入

这种录入方式要为每个单元格单独写入使用代码，适合对表格数据样式重复性低的情形，代码量大，可移植性差。

```
from docx import Document # 导入docx包
文件 = Document('c:/练习.docx') # 新建docx文档
表 = 文件.add_table(2, 4)
表.cell(0, 0).text = '序号'
表.cell(0, 1).text = '姓名'
表.cell(0, 2).text = '年龄'
表.cell(0, 3).text = '身高'
# 表格赋值，将第二行作为数据输入第一行
表.cell(1, 0).text = '1'
表.cell(1, 1).text = '孙兴华'
表.cell(1, 2).text = '20'
表.cell(1, 3).text = '178'
文件.save('c:/12.docx')
```

二、按行录入


这种方式将数据的录入以行为单位进行录入。在对数据录入时，先获取行中`cells`对象，遍历每个`cell`，并通过`cell.text`赋值，录入数据

```
from docx import Document # 导入docx包
数据 = [['序号', '姓名', '年龄', '身高'], ['1', '孙兴华', '20', '178'], ['2', '赵丽颖', '33', '165']]
文件 = Document('c:/练习.docx') # 新建docx文档
表 = 文件.add_table(3, 4)
表头 = 表.rows[0].cells
for 列 in range(4):
    表头[列].text = 数据[0][列] # ['序号', '姓名', '年龄', '身高']里面的第几个
# 录入数据
for 行 in range(1, 3):
    表数据 = 表.rows[行].cells
    for 列 in range(4): # 遍历列
        表数据[列].text = 数据[行][列] # 数据中每一行是一个列表，每一列是一个元素
文件.save('c:/13.docx')
```

★ 三、从表格中导入数据后按行录入

```
from docx import Document # 导入docx包
import pandas as pd
数据 = pd.read_excel('c:/表格.xlsx', header=None)
文件 = Document('c:/练习.docx')
表 = 文件.add_table(3, 4)
for 行 in range(3):
    for 列 in range(4):
        print(行, 列) # 可以查看表格输出结果
        表.cell(行, 列).text = str(数据.iloc[行, 列]) # 由于里面有数据型的，需要强制转字符串
文件.save('c:/13.docx')
```

```
import pandas as pd
数据 = pd.read_excel('c:/表格.xlsx', header=None)
print(数据)
print(数据.iloc[1, 3])
print(type(数据.iloc[1, 3]))
```



8.7 删除单元格数据

2020年6月21日 19:49

单元格数据的删除从本质上来说同赋值是一样的，只不过赋值为空字符串。同样也有两种方式来实现删除单元格数据，一种是直接cell.text赋值为空，另外一种能是使用段落，将段落的text属性赋值为空

```
from docx import Document # 导入docx包
```

```
文件 = Document('c:/13.docx')
```

```
表 = 文件.tables[0] # 读取第1个表格
```

```
# 第一种方法，对单元格赋值为空
```

```
表.cell(0,0).text = "" # 将表的第1行第1列单元格赋值为空
```

```
# 第二种方法，对段落赋值为空
```

```
段落 = 表.cell(0,1).paragraphs[0].text = "" # 表的第1行第2列第1个段落赋值为空
```

```
文件.save('c:/14.docx')
```

8.8 删除表格行或列数据

2020年6月21日 19:49

我个人在python-docx中没有找到删除一整行或列数据的方法，不过可以通过遍历行或列内的cell，然后利用删除行或列内所有单元格的方法来实现

一、删除表格行数据

```
from docx import Document # 导入docx包
文件 = Document('c:/13.docx')
表 = 文件.tables[0] # 读取第1个表格
for 单元格 in 表.rows[0].cells: # 遍历表格中第1行中所有的单元格
    单元格.text = "" # 将所有单元格赋值为空
文件.save('c:/14.docx')
```

二、删除表格列数据

```
from docx import Document # 导入docx包
文件 = Document('c:/13.docx')
表 = 文件.tables[0] # 读取第1个表格
for 单元格 in 表.columns[0].cells: # 遍历表格中第1列中所有的单元格
    单元格.text = "" # 将所有单元格赋值为空
文件.save('c:/14.docx')
```

8.9 表格对齐【调整列宽】

2020年6月22日 20:39

WD_TABLE_ALIGNMENT.LEFT	左对齐
WD_TABLE_ALIGNMENT.CENTER	居中对齐
WD_TABLE_ALIGNMENT.RIGHT	右对齐

一、表格对齐与列宽调整

```
from docx.enum.table import WD_TABLE_ALIGNMENT # 导入表格对齐方式
from docx.shared import Cm # 导入单位转换函数
from docx import Document
文件 = Document('c:/13.docx') # 新建docx文档
表 = 文件.tables[0] # 指定第1个表格
表.alignment = WD_TABLE_ALIGNMENT.LEFT # 设置表格为左对齐
for 列 in 表.columns: # 表格1设置列宽为10cm
    for 单元格 in 列.cells:
        单元格.width = Cm(1)
文件.save('c:/14.docx')
```

二、如果行高调整呢？

把columns改成rows

把列.cells改成行.cells

把单元格.width改成单元格.height

8.10 单元格对齐

2020年6月22日 20:39

在对单元格对齐方式设置的时候，将单元格视为一个整体，要使用单元格中的垂直对齐（cell.vertical_alignment）和单元格中的段落的对齐（paragraph.alignment）等2种对齐方式配合使用

在docx.enum.table.WD_ALIGN_VERTICAL定义了TOP、CENTER和BOTTOM等3种类型

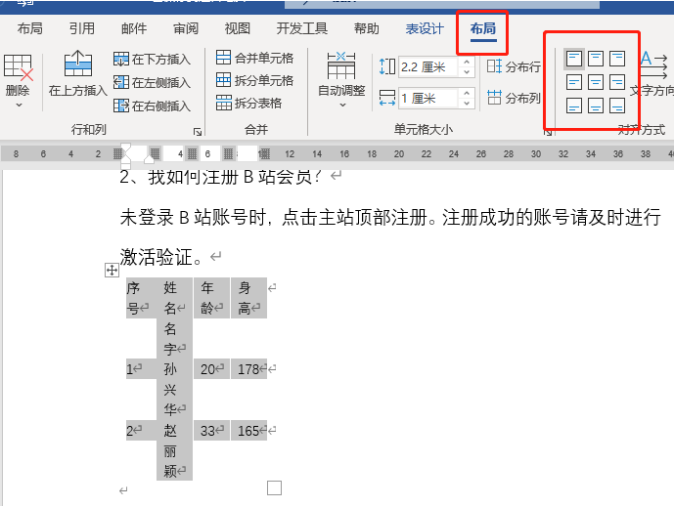
WD_CELL_VERTICAL_ALIGNMENT.TOP	单元格内容靠上对齐
WD_CELL_VERTICAL_ALIGNMENT.CENTER	单元格内容居中对齐
WD_CELL_VERTICAL_ALIGNMENT.BOTTOM	单元格内容靠下对齐

在WD_PARAGRAPH_ALIGNMENT中定义了4中类型，分别是LEFT、CENTER、RIGHT和JUSTIFY等4中类型

WD_PARAGRAPH_ALIGNMENT.LEFT	段落左对齐
WD_PARAGRAPH_ALIGNMENT.CENTER	段落居中对齐
WD_PARAGRAPH_ALIGNMENT.RIGHT	段落右对齐
WD_PARAGRAPH_ALIGNMENT.JUSTIFY	段落两端对齐

在单元格垂直对齐和段落对齐的配合过程中可以组合成12种方式，分别是：靠上两端对齐、靠上居中对齐、靠上右对齐、中部两端对齐、中部居中对齐、中部右对齐、靠下两端对齐、靠下居中对齐、靠下右对齐、靠上左对齐、中部左对齐、靠下左对齐。

其中，在WORD软件中内置了前9种对齐方式。



```
from docx import Document # 导入docx
from docx.enum.table import WD_CELL_VERTICAL_ALIGNMENT # 导入单元格垂直对齐
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT # 导入段落对齐
文件 = Document('c:/13.docx')
表 = 文件.tables[0] # 指定第1个表格
单元格 = 表.cell(0,1) # 指定单元格
# 靠上居中对齐
单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.TOP
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.CENTER
文件.save('c:/15.docx')
```

附：其它8种对齐方式

- 1、靠上两端对齐
单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.TOP
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.JUSTIFY
- 2、靠上右对齐
单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.TOP
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.RIGHT
- 3、中部两端对齐

单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.CENTER
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.JUSTIFY

4、中部居中对齐

单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.CENTER
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.CENTER

5、中部右对齐

单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.CENTER
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.RIGHT

6、靠下两端对齐

单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.BOTTOM
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.JUSTIFY

7、靠下中部对齐

单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.BOTTOM
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.CENTER

8、靠下右对齐

单元格.vertical_alignment = WD_CELL_VERTICAL_ALIGNMENT.BOTTOM
单元格.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.RIGHT

8.11 表格样式

2020年6月21日 21:05

```
from docx import Document
from docx.oxml.ns import qn # 中文字体
文件 = Document()
表 = 文件.add_table(3, 3, style="Medium Grid 1 Accent 1")
表头 = 表.rows[0].cells
表头[0].text = '姓名'
表头[1].text = '性别'
表头[2].text = '年龄'
# 将表格中所有单元格修改字体
for 行 in 表.rows:
    for 单元格 in 行.cells:
        for 段落 in 单元格.paragraphs:
            for 块 in 段落.runs:
                块.font.name = 'Arial' # 英文字体设置
                块._element.rPr.rFonts.set(qn('w:eastAsia'), '微软雅黑') # 设置中文字体

文件.save('c:/1.docx')
```

8.11.1 查看所有支持的表格样式

2020年6月23日 1:19

```
from docx.enum.style import WD_STYLE_TYPE # 读标题笔记05
from docx import Document # 可以写成 from docx import *
文件 = Document()
所有样式 = 文件.styles
#生成所有表样式
for 样式 in 所有样式:
    if 样式.type == WD_STYLE_TYPE.TABLE:
        文件.add_paragraph("表格样式 : "+ 样式.name)
        表 = 文件.add_table(3,3, style = 样式)
        单元格 = 表.rows[0].cells
        单元格[0].text = '第一列内容'
        单元格[1].text = '第二列内容'
        单元格[2].text = '第三列内容'
        文件.add_paragraph("\n")

文件.save('c:/1.docx') # 这个文档里就包含了所有支持的样式
```


附：表格样式列表

2020年6月23日 1:02

表格样式：Normal Table

第1列	第2列	第3列

表格样式：Table Grid

第1列	第2列	第3列

表格样式：Light Shading

第1列	第2列	第3列

表格样式：Light Shading Accent 1

第1列	第2列	第3列

表格样式：Light Shading Accent 2

第1列	第2列	第3列

表格样式：Light Shading Accent 3

第1列	第2列	第3列

表格样式：Light Shading Accent 4

第1列	第2列	第3列

表格样式：Light Shading Accent 5

第1列	第2列	第3列

表格样式：Light Shading Accent 6

第1列	第2列	第3列

表格样式：Light List

第1列	第2列	第3列

表格样式：Light List Accent 1

第1列	第2列	第3列

表格样式：Light List Accent 2

第1列	第2列	第3列

表格样式：Light List Accent 3

第1列	第2列	第3列

表格样式：Light List Accent 4

第1列	第2列	第3列

表格样式：Light List Accent 5

第1列	第2列	第3列

表格样式：Light List Accent 6

第1列	第2列	第3列

表格样式：Light Grid

第1列	第2列	第3列

表格样式：Light Grid Accent 1

第1列	第2列	第3列

表格样式：Light Grid Accent 2

第1列	第2列	第3列

表格样式：Light Grid Accent 3

第1列	第2列	第3列

表格样式：Light Grid Accent 4

第1列	第2列	第3列

表格样式：Light Grid Accent 5

第1列	第2列	第3列

表格样式：Light Grid Accent 6

第1列	第2列	第3列

表格样式：Medium Shading 1

第1列	第2列	第3列

表格样式：Medium Shading 1 Accent 1

第1列	第2列	第3列

表格样式：Medium Shading 1 Accent 2

第1列	第2列	第3列

表格样式：Medium Shading 1 Accent 3

第1列	第2列	第3列

表格样式：Medium Shading 1 Accent 4

第1列	第2列	第3列

表格样式：Medium Shading 1 Accent 5

第1列	第2列	第3列

表格样式：Medium Shading 1 Accent 6

第1列	第2列	第3列

表格样式：Medium Shading 2

第1列	第2列	第3列

表格样式：Medium Shading 2 Accent 1

第1列	第2列	第3列

表格样式：Medium Shading 2 Accent 2

第1列	第2列	第3列

表格样式：Medium Shading 2 Accent 3

第1列	第2列	第3列

表格样式：Medium Shading 2 Accent 4

第1列	第2列	第3列

表格样式：Medium Shading 2 Accent 5

第1列	第2列	第3列

表格样式：Medium Shading 2 Accent 6

第1列	第2列	第3列

表格样式：Medium List 1

第1列	第2列	第3列

表格样式：Medium List 1 Accent 1

第1列	第2列	第3列

表格样式：Medium List 1 Accent 2

第1列	第2列	第3列

表格样式：Medium List 1 Accent 3

第1列	第2列	第3列

表格样式：Medium List 1 Accent 4

第1列	第2列	第3列

表格样式：Medium List 1 Accent 5

第1列	第2列	第3列

表格样式：Medium List 1 Accent 6

第1列	第2列	第3列

表格样式：Medium List 2

第1列	第2列	第3列

表格样式：Medium List 2 Accent 1

第1列	第2列	第3列

表格样式：Medium List 2 Accent 2

第1列	第2列	第3列

表格样式：Medium List 2 Accent 3

第1列	第2列	第3列

表格样式：Medium List 2 Accent 4

第1列	第2列	第3列

表格样式：Medium List 2 Accent 5

第1列	第2列	第3列

表格样式：Medium List 2 Accent 6

第1列	第2列	第3列

表格样式：Medium Grid 1

第1列	第2列	第3列

表格样式：Medium Grid 1 Accent 1

第1列	第2列	第3列

表格样式：Medium Grid 1 Accent 2

第1列	第2列	第3列

表格样式：Medium Grid 1 Accent 3

第1列	第2列	第3列

表格样式：Medium Grid 1 Accent 4

第1列	第2列	第3列

表格样式：Medium Grid 1 Accent 5

第1列	第2列	第3列

表格样式：Medium Grid 1 Accent 6

第1列	第2列	第3列

表格样式：Medium Grid 2

第1列	第2列	第3列

表格样式：Medium Grid 2 Accent 1

第1列	第2列	第3列

表格样式：Medium Grid 2 Accent 2

第1列	第2列	第3列

表格样式：Medium Grid 2 Accent 3

第1列	第2列	第3列

表格样式：Medium Grid 2 Accent 4

第1列	第2列	第3列

表格样式：Medium Grid 2 Accent 5

第1列	第2列	第3列

表格样式：Medium Grid 2 Accent 6

第1列	第2列	第3列

表格样式：Medium Grid 3

第1列	第2列	第3列

表格样式：Medium Grid 3 Accent 1

第1列	第2列	第3列

表格样式：Medium Grid 3 Accent 2

第1列	第2列	第3列

表格样式：Medium Grid 3 Accent 3

第1列	第2列	第3列

表格样式：Medium Grid 3 Accent 4

第1列	第2列	第3列

表格样式：Medium Grid 3 Accent 5

第1列	第2列	第3列

表格样式：Medium Grid 3 Accent 6

第1列	第2列	第3列

表格样式：Dark List

第1列	第2列	第3列

表格样式：Dark List Accent 1

第1列	第2列	第3列

表格样式：Dark List Accent 2

第1列	第2列	第3列

表格样式：Dark List Accent 3

第1列	第2列	第3列

表格样式：Dark List Accent 4

第1列	第2列	第3列

表格样式：Dark List Accent 5

第1列	第2列	第3列

表格样式：Dark List Accent 6

第1列	第2列	第3列

表格样式：Colorful Shading

第1列	第2列	第3列

表格样式：Colorful Shading Accent 1

第1列	第2列	第3列

表格样式：Colorful Shading Accent 2

第1列	第2列	第3列

表格样式：Colorful Shading Accent 3

	第2列	第3列

表格样式：Colorful Shading Accent 4

第1列	第2列	第3列

表格样式：Colorful Shading Accent 5

第1列	第2列	第3列

表格样式：Colorful Shading Accent 6

第1列	第2列	第3列

表格样式：Colorful List

第1列	第2列	第3列

表格样式：Colorful List Accent 1

第1列	第2列	第3列

表格样式：Colorful List Accent 2

第1列	第2列	第3列

表格样式：Colorful List Accent 3

第1列	第2列	第3列

表格样式：Colorful List Accent 4

第1列	第2列	第3列

表格样式：Colorful List Accent 5

第1列	第2列	第3列

表格样式：Colorful List Accent 6

第1列	第2列	第3列

表格样式：Colorful Grid

第1列	第2列	第3列

表格样式：Colorful Grid Accent 1

第1列	第2列	第3列

表格样式：Colorful Grid Accent 2

第1列	第2列	第3列

表格样式：Colorful Grid Accent 3

第1列	第2列	第3列

表格样式：Colorful Grid Accent 4

第1列	第2列	第3列

表格样式：Colorful Grid Accent 5

第1列	第2列	第3列

表格样式：Colorful Grid Accent 6

第1列	第2列	第3列

09.文字样式调整

2020年6月20日 23:31

```
from docx import Document
from docx.shared import Pt, RGBColor # 字号, 颜色
from docx.oxml.ns import qn # 中文字体
文件 = Document('c:/练习.docx')
for 段落 in 文件.paragraphs:
    for 块 in 段落.runs:
        块.font.bold = True # 加粗
        块.font.italic = True # 斜体
        块.font.underline = True # 下划线
        块.font.strike = True # 删除线
        块.font.shadow = True # 阴影
        块.font.size = Pt(24)
        块.font.color.rgb = RGBColor(255,0,0) # 颜色
        块.font.name = 'Arial' # 英文字体设置
        块._element.rPr.rFonts.set(qn('w:eastAsia'),'微软雅黑') # 设置中文字体
文件.save('c:/练习6.docx')
```

对文字字体样式进行修改

run.font.样式 = xxx

附：文字样式调整方法

2020年6月21日 9:31

方法	作用
all_caps	全部大写字母
bold	加粗
color	字体颜色
complex_script	是否为“复杂代码”
cs_bold	“复杂代码”加粗
cs_italic	“复杂代码”斜体
double_strike	双删除线
emboss	文本以凸出页面的方式出现
hidden	隐藏
imprint	印记
italic	斜体
name	字体
no_proof	不验证语法错误
outline	显示字符的轮廓
shadow	阴影
small_caps	小型大写字母
snap_to_grid	定义文档网格时对齐网络
strike	删除线
subscript	下标
superscript	上标
underline	下划线

9.1 修改正文字体

2020年6月20日 23:55

```
from docx import Document
from docx.oxml.ns import qn # 中文字体
文件 = Document('c:/练习3.docx')
文件.styles['Normal'].font.name = 'Arial' # 设置英文字体
文件.styles['Normal']._element.rPr.rFonts.set(qn('w:eastAsia'), '微软雅黑') # 设置中文字体
文件.save('c:/1.docx')
```

注：文件.styles['Normal'] 意思是文件标题中的正文

9.2 修改标题字体

2020年6月20日 23:55

```
from docx import Document
from docx.oxml.ns import qn # 中文字体
文件 = Document('c:/练习3.docx')
文件.styles['Heading 1'].font.name = 'Arial'
文件.styles['Heading 1']._element.rPr.rFonts.set(qn('w:eastAsia'), '微软雅黑')
文件.save('c:/1.docx')
```

10.段落样式修改

2020年6月21日 1:22

对齐样式

paragraph.alignment = 对齐方式

10.1 对齐方式

2020年6月21日 1:24

LEFT	靠左
CENTER	居中
RIGHT	靠右
JUSTIFY	两端对齐
DISTRIBUTE	分散对齐

```
from docx.enum.text import WD_ALIGN_PARAGRAPH
```

```
from docx import Document
```

```
文件 = Document('c:/练习3.docx')
```

```
for 段落 in 文件.paragraphs:
```

```
    if 段落.style.name == 'Normal':
```

```
        段落.alignment = WD_ALIGN_PARAGRAPH.CENTER
```

```
文件.save('c:/1.docx')
```

选择标题或正文，详见笔记05

→ 对齐方式

LEFT，左对齐，值为0

CENTER，居中，值为1

RIGHT，右对齐，值为2

JUSTIFY，两端对齐，值为3

DISTRIBUTE，分散对齐，值为4

JUSTIFY_MED，以中等字符压缩比调整，值为5

JUSTIFY_HI，以高字符压缩比调整，值为6

JUSTIFY_LOW，以低字符压缩比调整，值为7

THAI_JUSTIFY，以泰语格式调整，值为8

10.2 行间距

2020年6月21日 9:13

行间距的设置可以使用2个属性`line_spacing`和`line_spacing_rule`。这两个属性不用同时设置。`line_spacing_rule`的值是`docx.enum.text.WD_LINE_SPACING`中的枚举类型的常量，值的列表如下：

`ONE_POINT_FIVE`，1.5倍行距

`AT_LEAST`，最小行距

`DOUBLE`，双倍行距

`EXACTLY`，固定值

`MULTIPLE`，多倍行距

`SINGL`，单倍行距

当`line_spacing_rule`的值设置为`EXACTLY`和`MULTIPLE`时，需要`line_spacing`属性进行设置具体的数值。

10.2.1 普通的行间距的调整

2020年6月21日 1:24

一、正文之间的行间距

```
from docx import Document
```

```
文件 = Document('c:/练习3.docx')
```

```
for 段落 in 文件.paragraphs:
```

```
    if 段落.style.name == 'Normal':
```

```
        段落.paragraph_format.line_spacing = 5.0
```

```
文件.save('c:/1.docx')
```

二、全部段落之间的行间距

```
from docx import Document
```

```
文件 = Document('c:/练习3.docx')
```

```
for 段落 in 文件.paragraphs:
```

```
    段落.paragraph_format.line_spacing = 5.0
```

```
文件.save('c:/1.docx')
```

10.2.2 特殊行间距调整

2020年6月21日 9:12

一、1.5倍行距

```
from docx.enum.text import WD_LINE_SPACING
from docx import Document
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    段落.paragraph_format.line_spacing_rule = WD_LINE_SPACING.ONE_POINT_FIVE
    print(段落.paragraph_format.line_spacing)
文件.save('保存路径')
```

二、固定值

```
from docx.enum.text import WD_LINE_SPACING
from docx import Document
from docx.shared import Pt
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    段落.line_spacing_rule = WD_LINE_SPACING.EXACTLY
    段落.paragraph_format.line_spacing = Pt(18)
文件.save('c:/10.docx')
```

三、多行倍距

```
from docx.enum.text import WD_LINE_SPACING
from docx import Document
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    段落.line_spacing_rule = WD_LINE_SPACING.MULTIPLE
    段落.paragraph_format.line_spacing = 1.75
文件.save('c:/10.docx')
```


10.3 段前与段后间距 【需要导入单位】

2020年6月21日 1:24

一、全部设置

```
from docx import Document
from docx.shared import Pt, Inches # Pt磅, Inches英寸
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    段落.paragraph_format.space_before = Pt(18) # 段前
    段落.paragraph_format.space_after = Pt(12) # 段后
文件.save('c:/1.docx')
```

二、设置正文部分

```
from docx import Document
from docx.shared import Pt, Inches # Pt磅, Inches英寸
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if 段落.style.name == 'Normal':
        段落.paragraph_format.space_before = Pt(18) # 段前
        段落.paragraph_format.space_after = Pt(12) # 段后
文件.save('c:/1.docx')
```

10.4 左缩进 【left_indent】

2020年6月21日 2:17

```
from docx import Document
from docx.shared import Pt, Inches # Pt磅, Inches英寸
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if 段落.style.name == 'Normal':
        段落.paragraph_format.left_indent=Inches(0.3) # 1 英寸=2.54
        厘米
文件.save('c:/1.docx')
```

pt	磅	int型
cm	厘米	float型
inches	英寸	float型
mm	毫米	float型

10.6 右缩进 【right_indent】

2020年6月21日 8:48

```
from docx import Document
from docx.shared import Pt, Inches # Pt磅, Inches英寸
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if 段落.style.name == 'Normal':
        段落.paragraph_format.right_indent=Inches(0.3) # 1 英寸=2.54 厘米
文件.save('c:/1.docx')
```

pt	磅	int型
cm	厘米	float型
inches	英寸	float型
mm	毫米	float型

10.5 首行缩进与悬挂缩进 【first_line_indent】

2020年6月21日 2:22

```
from docx import Document
from docx.shared import Pt, Inches # Pt磅, Inches英寸
文件 = Document('c:/练习3.docx')
for 段落 in 文件.paragraphs:
    if 段落.style.name == 'Normal':
        段落.paragraph_format.first_line_indent=Inches(0.3) # 1 英寸=2.54 厘米
文件.save('c:/1.docx')
```

首行缩进和悬挂缩进使用first_line_indent属性来实现，当值为大于0时，为首行缩进当值为小于0时为悬挂缩进

正数是首行缩进，负数是悬挂缩进，Inches(0.5)等于四个空格

11.节的添加、定位和分节符的设置

2020年6月21日 5:37

在WORD文档中添加节要使用“布局”菜单里“分隔符”按钮分节符功能

Word中节的作用是针对页面可以设置不一样的效果：

- 1、同一篇文档中可以通过节设置不同的纸张大小，方向以及页边距。
- 2、同一篇文档中可以通过节设置不同的页眉页脚，默认情况下所有页面均处于同一节，可以灵活设置多页是一节，也可以一页是就是一节。
- 3、同一篇文档中可以通过节控制页码的显示，主要针对文章带有目录，很多情况下，目录所在的页面是不占页码的，正文当第一页去使用。

```
from docx import Document # 导入docx
```

```
文件 = Document() # 新建一个文档
```

```
print('默认节的数量：', len(文件.sections)) # 打印默认节对数量
```

```
print('默认段落的数量：', len(文件.paragraphs)) # 打印默认段落数量
```

总结：一个节占用一个段落

11.1 添加节

2020年6月22日

21:33

```
from docx import Document # 导入docx
文件 = Document() # 新建一个文档
print('默认节的数量: ', len(文件.sections)) # 打印默认节对数量
print('默认段落的数量: ', len(文件.paragraphs)) # 打印默认段落数量
文件.add_section() # 添加第2个节
print('节的数量: ', len(文件.sections)) # 打印节对数量
print('段落的数量: ', len(文件.paragraphs)) # 打印段落数量
文件.add_section() # 添加第3个节
print('节的数量: ', len(文件.sections)) # 打印节对数量
print('段落的数量: ', len(文件.paragraphs)) # 打印段落数量
```

在添加分节符的时候，也添加了段落。现在再分别为这2个段落里添加文字

```
from docx import Document # 导入docx
文件 = Document() # 新建一个文档
print('默认节的数量: ', len(文件.sections)) # 打印默认节对数量
print('默认段落的数量: ', len(文件.paragraphs)) # 打印默认段落数量
文件.add_section() # 添加第2个节
print('节的数量: ', len(文件.sections)) # 打印节对数量
print('段落的数量: ', len(文件.paragraphs)) # 打印段落数量
文件.add_section() # 添加第3个节
print('节的数量: ', len(文件.sections)) # 打印节对数量
print('段落的数量: ', len(文件.paragraphs)) # 打印段落数量
文件.paragraphs[0].add_run('第1个段落')
文件.paragraphs[1].add_run('第2个段落')
文件.save('c:/18.docx')
```

```
文件.paragraphs[1].add_run('第2个段落')
```

```
文件.save('c:/18.docx')
```

11.2 节对定位

2020年6月22日 21:33

节 = 文件.sections[1] # 获取第2个节

11.3 分节符的设置

2020年6月22日 21:34

WD_SECTION_START.CONTINUOUS	连续分隔符
WD_SECTION_START.NEW_COLUMN	新列分隔符
WD_SECTION_START.NEW_PAGE	新页的分隔符
WD_SECTION_START.EVEN_PAGE	偶数页的分隔符
WD_SECTION_START.ODD_PAGE	奇数页的分隔符

```
from docx import Document
```

```
from docx.enum.section import WD_SECTION_START
```

```
文件 = Document('c:/18.docx')
```

```
节 = 文件.sections[1] # 获取第2个节, 用于更改分节符
```

```
print('更改前分节符类型: ', 节.start_type) # 更改分节符之前
```

```
节.start_type = WD_SECTION_START.ODD_PAGE # 设置奇数分节符
```

```
print('更改后分节符类型: ', 节.start_type) # 更改分节符之后
```

12.设置页眉和页脚

2020年6月21日 3:42

```
from docx import Document
```

```
文件 = Document('c:/练习7.docx')
```

```
页眉 = 文件.sections[0].header # 获取第一个节的页眉
```

```
print(f'页眉中的段落数: {len(页眉.paragraphs)}')
```

```
段落 = 页眉.paragraphs[0] # 获取页眉的第一个段落
```

```
段落.add_run('这是第一节的页眉') # 添加页面内容
```

```
页脚 = 文件.sections[0].footer # 获取第一个节的页脚
```

```
段落 = 页脚.paragraphs[0] # 获取页脚的第一个段落
```

```
段落.add_run('这是第一节的页脚') # 添加页脚内容
```

```
文件.save('c:/1.docx')
```

12.1 设置新的节

2020年6月21日 3:57

```
from docx import Document
```

```
文件 = Document('c:/练习7.docx')
```

```
页眉 = 文件.sections[0].header # 获取第一个节的页眉
```

```
print(f'页眉中的段落数: {len(页眉.paragraphs)}')
```

```
段落 = 页眉.paragraphs[0] # 获取页眉的第一个段落
```

```
段落.add_run('这是第一节的页眉') # 添加页面内容
```

```
页脚 = 文件.sections[0].footer # 获取第一个节的页脚
```

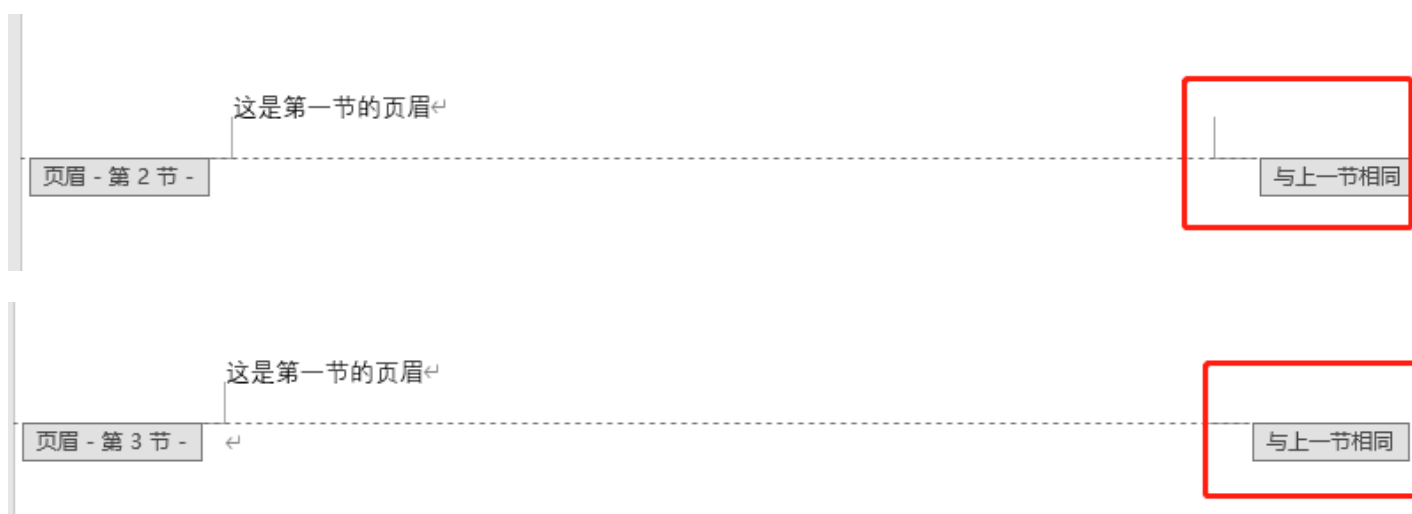
```
段落 = 页脚.paragraphs[0] # 获取页脚的第一个段落
```

```
段落.add_run('这是第一节的页脚') # 添加页脚内容
```

```
文件.add_section() # 添加一个新的节
```

```
文件.add_section() # 添加第3个节
```

```
文件.save('c:/1.docx')
```



12.2 指定某个节与其它设置不同

2020年6月21日 3:57



```
from docx import Document
```

```
文件 = Document('c:/练习7.docx')
```

```
页眉 = 文件.sections[0].header # 获取第一个节的页眉
```

```
print(f'页眉中的段落数: {len(页眉.paragraphs)}')
```

```
段落 = 页眉.paragraphs[0] # 获取页眉的第一个段落
```

```
段落.add_run('这是第一节的页眉') # 添加页面内容
```

```
页脚 = 文件.sections[0].footer # 获取第一个节的页脚
```

```
段落 = 页脚.paragraphs[0] # 获取页脚的第一个段落
```

```
段落.add_run('这是第一节的页脚') # 添加页脚内容
```

```
文件.add_section() # 添加一个新的节
```

```
文件.add_section() # 添加第3个节
```

```
页眉 = 文件.sections[1].header # 获取第2个节的页眉
```

```
页眉.is_linked_to_previous = False # 不使用上节内容和样式
```

```
文件.save('c:/1.docx')
```

12.3 设置对齐

2020年6月21日 3:57

页眉和页脚的内容是通过段落来添加的，所以通过设置段落的对齐方式就可以实现对页眉或者页脚的对齐。有关段落对齐以及更多的内容可以参考文章python-docx段落设置。以下代码分别对text.docx文档中的三个节分别设置对齐方式

```
from docx.enum.text import WD_ALIGN_PARAGRAPH
from docx import Document
文件 = Document('c:/1.docx')
页眉 = 文件.sections[1].header # 获取第2个节的页眉
页眉.is_linked_to_previous = False # 不使用上节内容和样式
段落 = 页眉.paragraphs[0] # 获取页眉的第一个段落
段落.add_run('这是第二节的页眉')
段落.alignment = WD_ALIGN_PARAGRAPH.CENTER # 设置页眉居中对齐
页脚 = 文件.sections[1].footer # 获取第2个节的页脚
页脚.is_linked_to_previous = False # 不使用上节内容和样式
段落 = 页脚.paragraphs[0] # 获取页脚的第一个段落
段落.add_run('这是第二节的页脚')
段落.alignment = WD_ALIGN_PARAGRAPH.CENTER # 设置页脚居中对齐
文件.save('c:/2.docx')
```

可以将部分内容写成一行：

```
from docx.enum.text import WD_ALIGN_PARAGRAPH
from docx import Document
文件 = Document('c:/1.docx')
页眉 = 文件.sections[1].header
页脚 = 文件.sections[1].footer
页眉.is_linked_to_previous = False # 获取第2个节的页眉，不使用上节内容和样式
页眉.paragraphs[0].add_run('这是第二节的页眉') # 获取页眉的第一个段落，加入文字
页眉.paragraphs[0].alignment = WD_ALIGN_PARAGRAPH.CENTER # 设置页眉居中对齐

页脚.is_linked_to_previous = False # 获取第2个节的页脚，不使用上节内容和样式
页脚.paragraphs[0].add_run('这是第二节的页脚') # 获取页脚的第一个段落，加入文字
页脚.paragraphs[0].alignment = WD_ALIGN_PARAGRAPH.CENTER # 设置页脚居中对齐
文件.save('c:/2.docx')
```

LEFT	靠左
CENTER	居中
RIGHT	靠右
JUSTIFY	两端对齐
DISTRIBUTE	分散对齐

12.4 页眉和页脚距离

2020年6月21日 4:37

在section中的:

header_distance属性可以用来设置页眉距离顶端的距离,
footer_distance属性则用来设置页脚距离底端的距离

```
from docx.shared import Cm # 导入单位转换
```

```
from docx import Document
```

```
文件 = Document('c:/2.docx')
```

```
文件.sections[1].header_distance = Cm(10) # 设置页眉为10厘米
```

```
文件.sections[1].footer_distance = Cm(10) # 设置页脚为10厘米
```

```
文件.save('c:/3.docx')
```

12.5 奇偶页不同

2020年6月21日 4:49

在python-docx中可以使用document.settings中的odd_and_even_pages_header_footer属性设置奇偶页不同

```
from docx import Document
```

```
文件 = Document('c:/2.docx')
```

```
print('默认文档奇偶页设置: ',文件.settings.odd_and_even_pages_header_footer)
```

```
文件.settings.odd_and_even_pages_header_footer = True # 启动奇偶页不同
```

```
文件.save('c:/3.docx')
```

在docx文档中odd_and_even_pages_header_footer属性默认为False，即文档的奇偶页是相同的。当设置为True时，启动了奇偶页不同。在python-docx中含有的偶数页页眉对象section.even_page_header，偶数页页脚对象section.even_page_footer；但却没有奇数页的对象，当设置的section.header或者section.footer时，默认设置在奇数页上。

```
from docx.enum.text import WD_ALIGN_PARAGRAPH
```

```
from docx.shared import Pt # 导入单位转换
```

```
from docx import Document
```

```
文件 = Document('c:/2.docx')
```

```
文件.settings.odd_and_even_pages_header_footer = True # 启动奇偶页不同
```

```
第一个节 = 文件.sections[0] # 第一个节
```

```
偶数页 = 第一个节.even_page_header # 获取偶数页
```

```
偶数页眉 = 偶数页.paragraphs[0].add_run('这是偶数页页眉')
```

```
偶数页眉.font.size = Pt(26) # 设置偶数页页眉大小
```

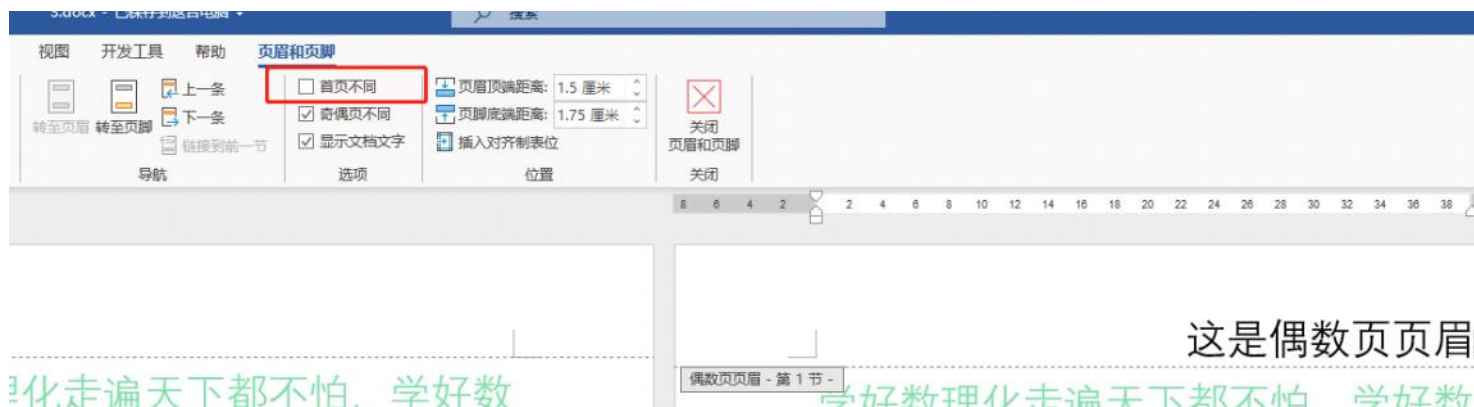
```
偶数页.paragraphs[0].alignment = WD_ALIGN_PARAGRAPH.RIGHT # 设置页眉右对齐
```

```
文件.save('c:/3.docx')
```

注：页脚同样的方法去设置

12.6 首页不同

2020年6月21日 5:10



设置首页不同要用到section中的
different_first_page_header_footer对象，对应于WORD软件位置，见上图。

```
from docx import Document
```

```
文件 = Document('c:/3.docx')
```

```
第1个节 = 文件.sections[0] # 获取第1个节的对象
```

```
print('首页不同默认状态: ', 第1个节.different_first_page_header_footer)
```

```
第1个节.different_first_page_header_footer = True # 设置启动首页不同
```

```
文件.save('c:/4.docx')
```

注：这时首页的没有页眉和页脚了，现在的首页被分成了首页和非首页两个部分

```
from docx.enum.text import WD_ALIGN_PARAGRAPH
```

```
from docx import Document
```

```
from docx.shared import Pt
```

```
文件 = Document('c:/3.docx')
```

```
第1个节 = 文件.sections[0] # 获取第1个节的对象
```

```
第1个节.different_first_page_header_footer = True # 设置启动首页不同
```

```
首页对象 = 第1个节.first_page_header # 先要启动首页不同，才能设置对象
```

```
首页页眉 = 首页对象.paragraphs[0].add_run('这是首页')
```

```
首页页眉.font.size = Pt(48)
```

```
首页对象.paragraphs[0].alignment = WD_ALIGN_PARAGRAPH.LEFT
```

```
文件.save('c:/4.docx')
```

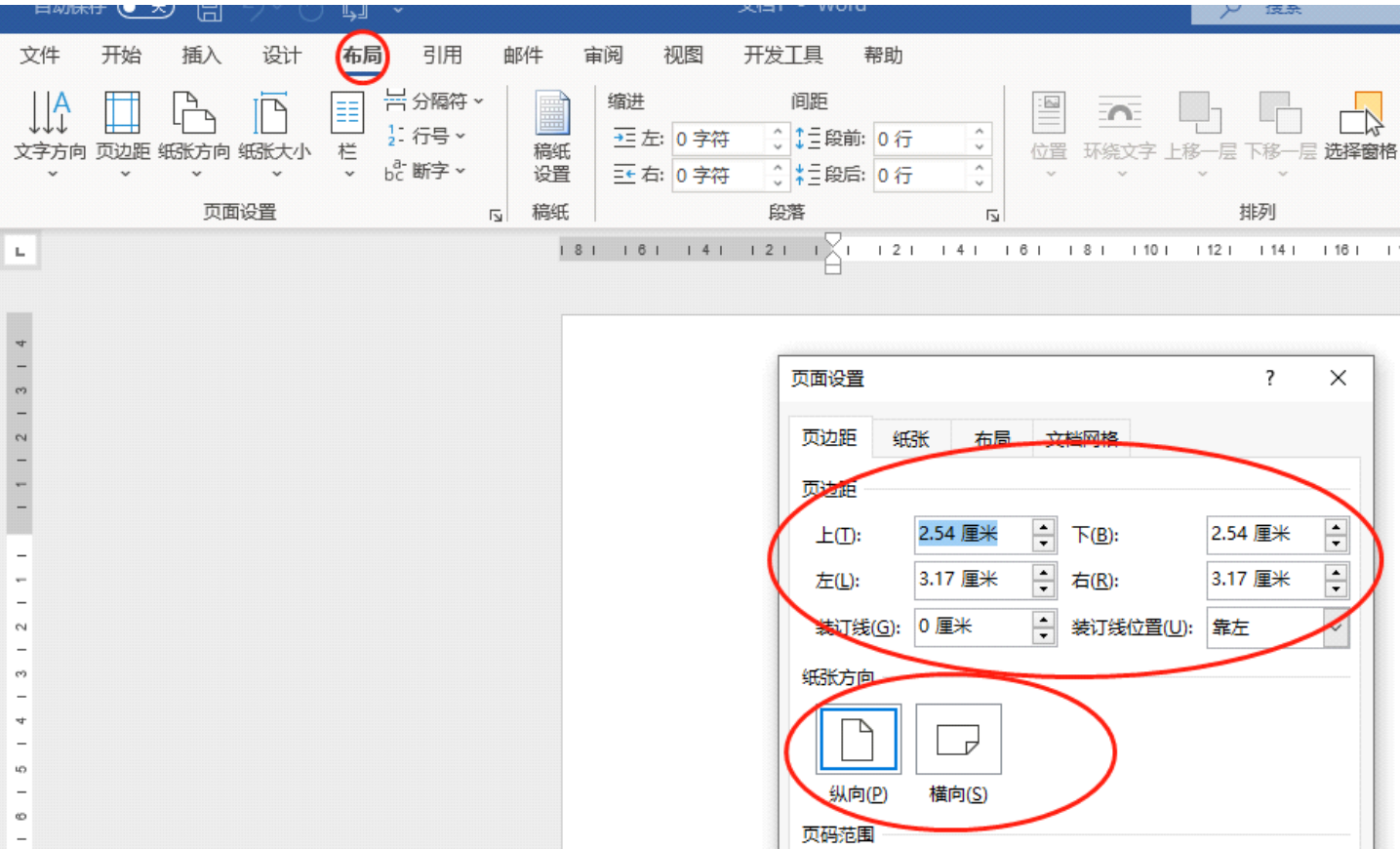

12.7 优先级

2020年6月21日 5:19

优先级生效的顺序依次是**首页不同>奇偶页>普通页**

13.页面设置

2020年6月21日 5:31



13.1 页面大小

2020年6月23日 1:29



```
from docx import Document
```

```
from docx.shared import Cm
```

```
文件 = Document() # 新建文档
```

```
第1个节 = 文件.sections[0] # 获取第1个节
```

```
# 打印默认页面宽度和高度
```

```
print('默认页面的宽度和高度: ', 第1个节.page_width.cm,第1个节.page_height.cm)
```

```
第1个节.page_width = Cm(40)
```

```
第1个节.page_height = Cm(40)
```

```
# 打印修改后的页面宽度和高度
```

```
print('修改后页面的宽度和高度: ', 第1个节.page_width.cm,第1个节.page_height.cm)
```

```
文件.save('c:/1.docx')
```

13.2 设置纸张

2020年6月23日 1:29

纸张大小(R):

A4	▼
宽度(W):	21 厘米
高度(E):	29.7 厘米

纸张大小(R):

B5	▼
宽度(W):	18.2 厘米
高度(E):	25.7 厘米

```
from docx import Document
```

```
from docx.shared import Cm
```

```
文件 = Document() # 新建文档
```

```
第1个节 = 文件.sections[0] # 获取第1个节
```

```
第1个节.page_width = Cm(21) # 宽
```

```
第1个节.page_height = Cm(29.7) # 高
```

```
文件.save('c:/1.docx')
```

13.3 设置纸张方向

2020年6月23日 1:29

WD_ORIENTATION.LANDSCAPE	纸张方向为横向
WD_ORIENTATION.PORTRAIT	纸张方向为纵向[默认]

一、针对新文档

```
from docx.enum.section import WD_ORIENTATION # 纸张方向
from docx import Document
from docx.shared import Cm
文件 = Document() # 新建文档
第1个节 = 文件.sections[0] # 获取第1个节
第1个节.orientation = WD_ORIENTATION.LANDSCAPE # 指定为横向, 不能省略, 否则尺寸变了但还是纵向
第1个节.page_width = Cm(29.7) # 宽
第1个节.page_height = Cm(21) # 高
文件.save('c:/1.docx')
```

二、针对已有文档

```
from docx.enum.section import WD_ORIENTATION # 纸张方向
from docx import Document
from docx.shared import Cm
文件 = Document('c:/练习.docx') # 新建文档
第1个节 = 文件.sections[0] # 获取第1个节
第1个节.orientation = WD_ORIENTATION.LANDSCAPE # 指定为横向
宽, 高 = 第1个节.page_width, 第1个节.page_height
第1个节.page_width = 高 # 宽=高
第1个节.page_height = 宽 # 高=宽
文件.save('c:/1.docx')
```

13.4 设置页面边距

2020年6月23日 1:30

<code>section.top_margin</code>	上边距
<code>section.bottom_margin</code>	下边距
<code>section.left_margin</code>	左边距
<code>section.right_margin</code>	右边距

```
from docx import Document
```

```
from docx.shared import Cm
```

```
文件 = Document('c:/练习.docx') # 新建文档
```

```
第1个节 = 文件.sections[0] # 获取第1个节
```

```
print("上边距: ", 第1个节.top_margin.cm)
```

```
print("下边距: ", 第1个节.bottom_margin.cm)
```

```
print("左边距: ", 第1个节.left_margin.cm)
```

```
print("右边距: ", 第1个节.right_margin.cm)
```

```
第1个节.top_margin = Cm(5)
```

```
第1个节.bottom_margin = Cm(5)
```

```
第1个节.left_margin = Cm(4)
```

```
第1个节.right_margin = Cm(4)
```

修改上下左右边距



```
print("上边距: ", 第1个节.top_margin.cm)
```

```
print("下边距: ", 第1个节.bottom_margin.cm)
```

```
print("左边距: ", 第1个节.left_margin.cm)
```

```
print("右边距: ", 第1个节.right_margin.cm)
```

```
文件.save('c:/1.docx')
```

13.6 设置装订线

2020年6月23日 1:30

```
from docx import Document
```

```
from docx.shared import Cm
```

```
文件 = Document('c:/练习.docx') # 新建文档
```

```
第1个节 = 文件.sections[0] # 获取第1个节
```

```
print('设置装订前', 第1个节.gutter.cm)
```

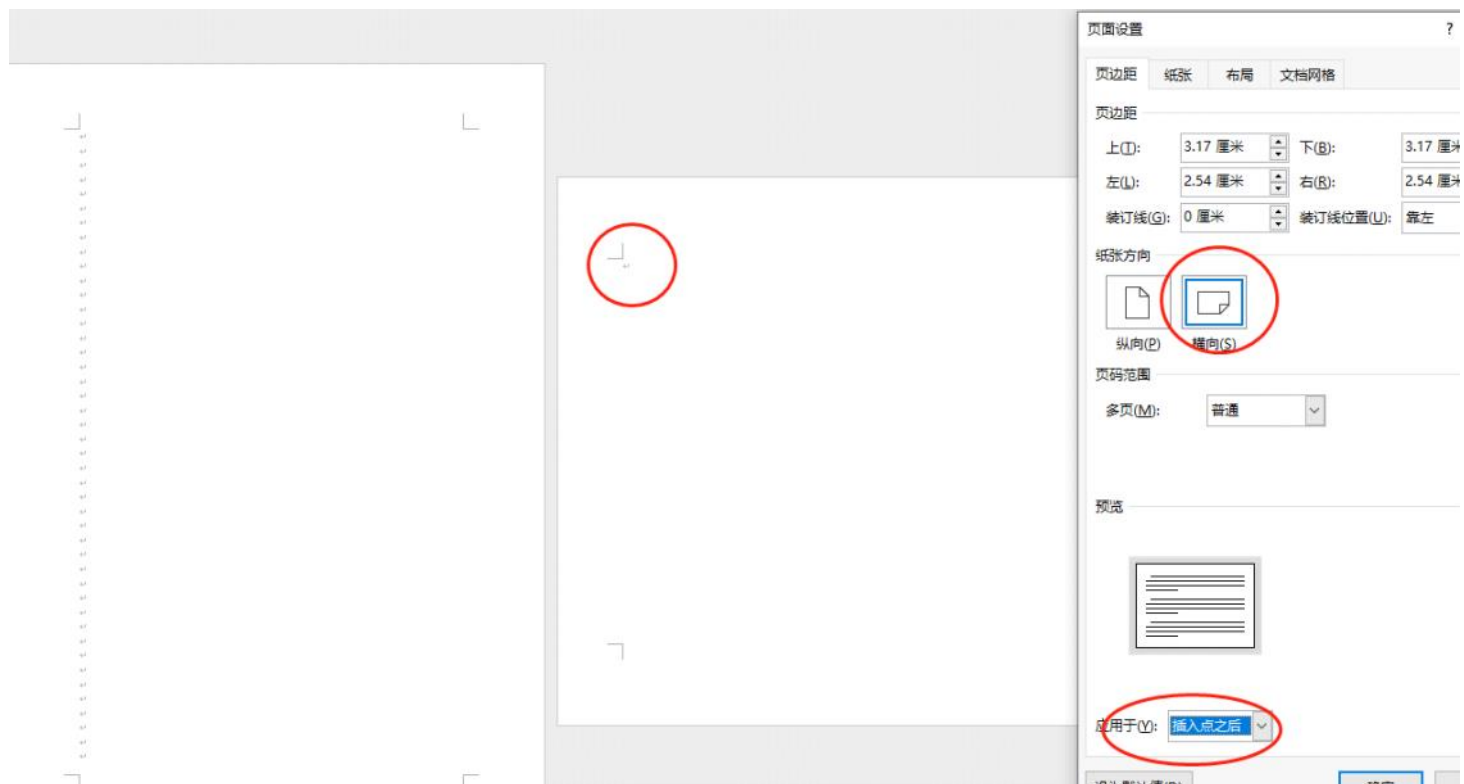
```
第1个节.gutter = Cm(1)
```

```
print('设置装订线后: ', 第1个节.gutter.cm)
```

```
文件.save('c:/1.docx')
```

13.7 WORD软件中设置部分页面为横向

2020年6月21日 5:37



到了第3页时，再重复一次就可以对第三页设定了

13.8 使用代码设置部分页面为横向

2020年6月23日 2:30

切记：文本甚用！

```
from docx import Document
from docx.enum.section import WD_ORIENTATION, WD_SECTION_START # 导入节方向和分页符类型
文件 = Document() # 新建docx文档
文件.add_paragraph() # 添加一个空白段落
节 = 文件.add_section(start_type=WD_SECTION_START.CONTINUOUS) # 添加横向页的连续分页符 笔记11.3
节.orientation = WD_ORIENTATION.LANDSCAPE # 设置横向
宽, 高 = 节.page_width, 节.page_height
节.page_width = 高 # 宽=高
节.page_height = 宽 # 高=宽
文件.add_paragraph() # 添加第二个空白段落
节 = 文件.add_section(start_type=WD_SECTION_START.CONTINUOUS) # 添加连续分页符
节.orientation = WD_ORIENTATION.PORTRAIT # 设置纵向
宽, 高 = 节.page_width, 节.page_height # 读取插入节的高和宽
节.page_width = 高 # 宽=高
节.page_height = 宽 # 高=宽
文件.save('c:/1.docx')
```

14.word批量转PDF

2020年6月22日 21:00

1、找到pip3.exe所在的文件夹，复制路径

我的路径是： C:\Users\孙艺航\AppData\Local\Programs\Python\Python37\Scripts

2、按Win+R,输入CMD确定

3、进入后，先输入cd 路径 回车

4、输入 pip3 install comtypes 回车

附送代码：

2020年6月22日 22:15

```
import os
```

```
import comtypes.client
```

```
def get_path():
```

```
    # 指定路径
```

```
    path = 'C:/word'
```

```
    # 获取所有文件名的列表
```

```
    filename_list = os.listdir(path)
```

```
    # 获取所有word文件名列表
```

```
    wordname_list = [filename for filename in filename_list \
                      if filename.endswith((".doc", ".docx"))]
```

```
    for wordname in wordname_list:
```

```
        # 分离word文件名称和后缀，转化为pdf名称
```

```
        pdfname = os.path.splitext(wordname)[0] + '.pdf'
```

```
        # 如果当前word文件对应的pdf文件存在，则不转化
```

```
        if pdfname in filename_list:
```

```
            continue
```

```
        # 拼接 路径和文件名
```

```
        wordpath = os.path.join(path, wordname)
```

```
        pdfpath = os.path.join(path, pdfname)
```

```
        #生成器
```

```
        yield wordpath,pdfpath
```

```
def convert_word_to_pdf():
```

```
    word = comtypes.client.CreateObject("Word.Application")
```

```
    word.Visible = 0
```

```
    for wordpath,pdfpath in get_path():
```

```
        newpdf = word.Documents.Open(wordpath)
```

```
        newpdf.SaveAs(pdfpath, FileFormat=17)
```

```
        newpdf.Close()
```

```
if __name__ == "__main__":
```

```
    convert_word_to_pdf()
```

→修改你的路径，把word文件都放在里面

综合练习：复工证明

2020年6月23日 0:42

```
from docx import Document
from docx.enum.text import WD_ALIGN_PARAGRAPH #对齐
from docx.shared import Pt, Inches # 单位
from docx.oxml.ns import qn # 中文字体
文件 = Document() # 新建文档
import pandas as pd
数据 = pd.read_excel('c:/复工证明统计表.xlsx')
# print(数据.iloc[0,1])
import datetime as dt
日期 = dt.datetime.now()
当前日期 = str(日期.year) + '年' + str(日期.month) + '月' + str(日期.day) + '日'
# print(当前日期)

for i in 数据['序号']: # 笔记8.6
    姓名 = 数据.iloc[i-1,1]
    身份证 = str(数据.iloc[i-1,2])
    电话 = str(数据.iloc[i-1,3])
    住址 = str(数据.iloc[i-1,4])
    来自 = str(数据.iloc[i-1,5])

    段落1 = 文件.add_paragraph() # 增加一个段落，这个段落是标题
    段落1.alignment = WD_ALIGN_PARAGRAPH.CENTER # 对齐方式为居中，没有这句话默认左对齐
    标题 = 段落1.add_run('复工证明') # 给段落添加一个块，写上文字
    标题.font.name = 'Arial' # 设置英文字体
    标题.element.rPr.rFonts.set(qn('w:eastAsia'), '黑体') # 设置中文字体
    标题.font.size = Pt(18) # 设置字号
    标题.font.bold = True # 设置加粗
    段落1.space_after = Pt(5) # 断后距离5磅
    段落1.space_before = Pt(5) # 断前距离5磅

    段落2 = 文件.add_paragraph() # 增加第二个段落，这个段落是正文
    正文 = 段落2.add_run(f'兹有{大唐天子圣谕, {姓名}同志, 身份证号(身份证), 联系电话(电话)}, 系我天朝(住址)子民, 根据我天朝复工要求, 该同志在防疫管控期间需要保护唐僧赴西天大雷音寺求取真经, 在满足(来自)当地疫情管控要求情况下, 请检测体温无异常后予以放行。')
    正文.font.name = 'Arial'
    正文.element.rPr.rFonts.set(qn('w:eastAsia'), '黑体')
    正文.font.size = Pt(14)
    正文.font.bold = True
    段落2.paragraph_format.first_line_indent = Inches(0.4) # 左缩进0.4英寸

    段落3 = 文件.add_paragraph() # 增加一个自然段，这段是特此说明
    说明 = 段落3.add_run('特此说明。')
    说明.font.name = 'Arial'
    说明.element.rPr.rFonts.set(qn('w:eastAsia'), '黑体')
    说明.font.size = Pt(14)
    说明.font.bold = True
    段落3.paragraph_format.first_line_indent = Inches(0.4)

    段落4 = 文件.add_paragraph() # 增加一个自然段，这段是单位名
    单位 = 段落4.add_run('大唐天子李世民')
    单位.font.name = 'Arial'
    单位.element.rPr.rFonts.set(qn('w:eastAsia'), '黑体')
```

```
单位.font.size = Pt(14)
单位.font.bold = True
段落4.paragraph_format.first_line_indent = Inches(4.0)
```

```
段落5 = 文件.add_paragraph() # 增加一个自然段
日期 = 段落5.add_run(f'{当前日期}')
日期.font.name = 'Arial'
日期.element.rPr.rFonts.set(qn('w: eastAsia'), '黑体')
日期.font.size = Pt(14)
日期.font.bold = True
段落5.paragraph_format.first_line_indent = Inches(3.9)
```

```
文件.add_page_break() # 增加分页符
```

```
文件.save('c:/复工证明.docx')
```

复工证明↵

兹有大唐天子圣谕，**孙悟空**同志，身份证号 **1001**,联系电话 **123**，系我天朝**花果山**子民，根据我天朝复工要求，该同志在防疫管控期间需要保护唐僧赴西天大雷音寺求取真经，在满足**凌霄宝殿**当地疫情管控要求情况下，请检测体温无异常后予以放行。↓

↵

特此说明。↓

↓

↓

↵

大唐天子李世民↓

↵

2020 年 6 月 23 日↵

↵