Word 神器 python-docx

原创 太阳雪 Python技术 1周前

前两天有个朋友向我求助，她在写毕业论文时，不小心将论文里的中文双引号替换为英文的了，各种原因导致无法回退，8万多字的论文，眼看就要交了，该怎么办？

首先想到 word 自身的替换功能，倒是能查到，但是没法动态替换，即只替换两边引号，而不换中间内容；

另外一种方案是，即用 VBA，通过编程来替换，虽说做过几个项目，可好久不用，拾起费劲，再加上 VBA 中各种概念和用法，学习成本太高，放弃；

还有一种方案，即用 Python 操作 word，首先对 Python 更熟悉，另外一定有别人造好的轮子。果然，没用多久找到了 python-docx Python 库，文档齐全，功能强大，用来解决替换问题不在话下。

开始之前，先简单了解下 python-docx

python-docx 介绍

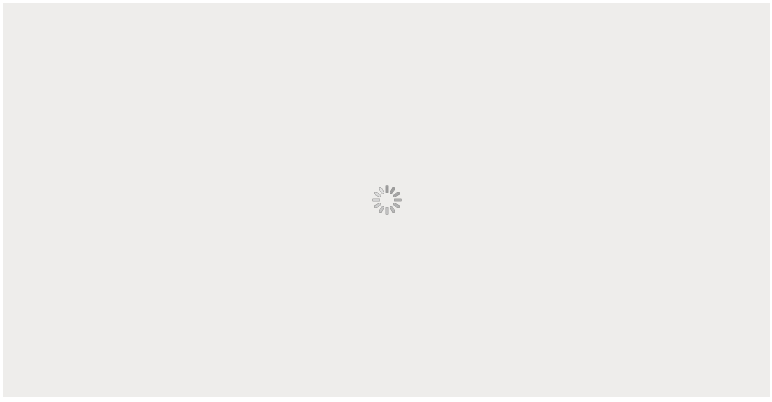
python-docx 是用于创建可修改 微软 Word 的一个 python 库，提供全套的 Word 操作，是最常用的 Word 工具

概念

使用前，先了解几个概念：

- `Document`：是一个 Word 文档 对象，不同于 VBA 中 Worksheet 的概念，Document 是独立的，打开不同的 Word 文档，就会有不同的 Document 对象，相互之间没有影响
- `Paragraph`：是段落，一个 Word 文档由多个段落组成，当在文档中输入一个回车键，就会成为新的段落，输入 shift + 回车，不会分段
- `Run` 表示一个节段，每个段落由多个 节段 组成，一个段落中具有相同样式的连续文本，组成一个节段，所以一个 段落 对象有个 Run 列表

例如有一个 Word，内容是：



则 结构这样划分：

第二个 段落（paragraph），没有内容，所以 节段（run）为空

安装

可以用 pip 来安装：

```
pip install python-docx
```

命令行中运行下面语句，如果没有报错，则说明安装成功

```
$ python -c 'import docx'
```

小试牛刀

python-docx 安装后，测试一下：

```
from docx import Document

document = Document()
paragraph = document.add_paragraph('Lorem ipsum dolor sit amet.')
prior_paragraph = paragraph.insert_paragraph_before('Lorem ipsum')

document.save(r"D:\test.docx")
```

- 引入 Document 类
- 定义一个新文档对象 document
- 想文档中插入一个段落（paragraph）
- 再在这个段落（paragraph）前插入另一个段落
- 最后调用文档对象 document 的 save 保存文档

用 Word 打开保存的 test.docx 就可以看到：

问题分析与解决

了解了 python-docx 的基本概念，开始着手解决问题，大体思路是：

1. 读取文档内容

2. 查找 英文引号 之间的内容
3. 将找到的内容的 英文引号 换成 中文引号，并将内容替换回去
4. 完成处理后将文档另存

查找目标

首先要解决的是如何找到 英文引号之间的内容？

例如文档内容有这么一段：

```
...
对"基于需求的教育资源配
置系统观"的研究，尤其是对"以学习者为中心"和从"个性化学习"、"精准教学"视角出发的
教育资源配置问题提供了理论"支持\\以及"方向指导
...
```

对于英文引号来说不区分前引号和后引号，怎么能保证配置到的不会是 `"和从"`、`"、"` 以及 `"以学习者为中心"和从"个性化学习"、"精准教学"` 或者 不会忽略两个引号出现在上下行的情况？

重温正则表达式，终于得到如下表达式：

```
'"(?:[^"])*"'
```

- `?:`：为了取消圆括号模式配置过程的缓存，即不需要遇到一个符合的就结束匹配
- `[^"]`：表示匹配的内容不能是 `"`，以避免贪婪匹配，即避免匹配成 从第一个 `"` 开始一直到最后一个 `"` 结束
- 整体的意思是 配置两个 `"` 之间的内容，且内容中不包括 `"`

后来整理过程中，还发现另一种写法：

```
'".*?"'
```

不过 `.` 不能匹配换行符 `\n`，坚持要用，需要使用 `可选修饰符` `re.S`：

```
import re
pattern = re.compile('".*?"', re.S)

re.findAll(pattern, text)  # text 为待查找字符串
```

- 引入 正则表达式模块 `re`
- `re.S` 为可选标识修饰符，使.匹配包括换行在内的所有字符
- 利用 `findAll` 查找所有匹配内容

> 关于 Python 正在表达式的更多用法参考文后参考链接

实现

查找问题解决了，做替换就方便多了：

```
from docx import Document
import re

doc = Document(r"D:\论文.docx")
restr = '"(?:[^"])*"'

for p in doc.paragraphs:
    matchRet = re.findall(restr, p.text)
    for r in matchRet:
        p.text = p.text.replace(r, '“' + r[1:-1] + '”')
doc.save(r'D:\论文_修正.docx')
```

- 引入 Document 类，和正则表达式模块
- 打开目标文档，字符串前的 `r` 表示取消字符串转义，即按原始字符产来解释
- 循环文档的 段落（paragraph），对每个段落，用正则表达式进行匹配
- 循环对于匹配到的结果，将前后引号，换成中文引号，并替换 段落（paragraph）的 `text`；其中 `r[1:-1]` 表示截取从第二个位置(第一个位置是 0）到倒数第二个位置截取字符串，刚好去掉前后引号
- 最后另存文档

> 注意：python-docx 保存文档时不会给出任何提示，会瞬间完成，所以另存是个稳妥的做法

完工，赶紧将替换好的文档发过去……

还没来得回味，她说：“非常感谢！那个～ 能不能再帮我生成个图表目录，这个必须要……”

好吧，能者多劳（神器在手），干就完了……

强大的 python-docx

在上面小试牛刀中，介绍了插入段落（paragraph）的用法，下面在介绍一些 python-docx 的其他功能

> 为了简洁，下面例子中省略了 Document 类的引入和实例化代码，document 为 Document 的实例

添加标题

默认情况下添加的标题是最高一级的，即一级标题，通过参数 `level` 设定，范围是 1～9，也有 0 级别，表示的是段

落标题：

```
# 添加一级标题
document.add_heading('我是一级标题')

decument.add_heading('我是二级标题', level=2)

decument.add_heading('我是段落标题', level=0)
```

添加换页

如果一个段落不满一页，需要分页时，可以插入一个分页符，直接调用会将分页符插入到最后一个段落之后:

```
# 文档最后插入分页
document.add_page_break()

# 特定段落分页
from docx.enum.text import WD_BREAK
paragraph = document.add_paragraph("独占一页")  # 添加一个段落
paragraph.runs[-1].add_break(WD_BREAK.PAGE)  # 在段落的最后一个节段后添加分页
```

表格操作

Word 文档中经常会用到表格，python-docx 如何添加和操作表格呢？

```
# 添加一个 2×2 表格
table = document.add_table(rows=2, cols=2)

# 获取第一行第二列单元格
cell = table.cell(0, 1)

# 设置单元格文本
cell.text = '我是单元格文字'

# 表格的行
row = table.rows[1]
row.cells[0].text = 'Foo bar to you.'
row.cells[1].text = 'And a hearty foo bar to you too sir!'

# 增加行
row = table.add_row()
```

更复杂点的例子：

```
# 表格数据
items = (
```

```
    (7, '1024', '手机'),
    (3, '2042', '笔记本'),
    (1, '1288', '台式机'),
)

# 添加一个表格
table = document.add_table(1, 3)

# 设置表格标题
heading_cells = table.rows[0].cells
heading_cells[0].text = '数量'
heading_cells[1].text = '编码'
heading_cells[2].text = '描述'

# 将数据填入表格
for item in items:
    cells = table.add_row().cells
    cells[0].text = str(item[0])
    cells[1].text = item[1]
    cells[2].text = item[2]
```

添加图片

添加图片，即，为 Word 里 菜单中 插入 > 图片 插入的功能，插入图片为原始大小：

```
document.add_picture('image-filename.png')
```

插入时设置图片大小：

```
from docx.shared import Cm
# 设置图片的跨度为 10 厘米
document.add_picture('image-filename.png', width=Cm(10))
```

> 除了厘米，python-docx 还提供了 英寸（Inches），如设置 1英寸: `Inches(1.0)`

样式

样式可以针对整体文档（document）、段落（paragraph）、节段（run），月具体，样式优先级越高

python-docx 样式功能配置多样，功能丰富，这里对段落样式和文字样式做简单介绍

段落样式

段落样式包括：对齐、列表样式、行间距、缩进、背景色等，可以在添加段落时设定，也可以在添加之后设置：

```
# 添加一个段落，设置为无序列表样式
document.add_paragraph('我是个无序列表段落', style='List Bullet')

# 添加段落后，通过 style 属性设置样式
paragraph = document.add_paragraph('我也是个无序列表段落')
paragraph.style = 'List Bullet'
```

文字样式

在前面 python-docx 文档结构图可以看到，段落中，不同样式的内容，被划分成多个 节段（Run），文字样式是通过 节段（Run）来设置的

设置加粗/斜体

```
paragraph = document.add_paragraph('添加一个段落')
# 设置 节段文字为加粗
run = paragraph.add_run('添加一个节段')
run.bold = True

# 设置 节段文字为斜体
run = paragraph.add_run('我是斜体的')
run.italic = True
```

设置字体

设置字体稍微复杂些，例如设置一段文字为 `宋体`：

```
paragraph = document.add_paragraph('我的字体是 宋体')
run = paragraph.runs[0]
run.font.name = '宋体'
run._element.rPr.rFonts.set(qn('w:eastAsia'), '宋体')
```

总结

python-docx 是个功能强大的 Word 库，能实现几乎所有在 Word 中操作，今天通过一个实例，介绍了 python-docx 的一些基本用法，限于篇幅，没法展开讨论更多内容，如果有兴趣可以深入研究，说不定可以让 Word 像 Markdown 一样简单。

参考

https://python-docx.readthedocs.io/en/latest/

https://www.runoob.com/python/python-reg-expressions.html

https://www.cnblogs.com/nixindecat/p/12157623.html

【代码获取方式】

识别文末二维码，回复：666

PS： 公号内回复「Python」即可进入 Python 新手学习交流群，一起 **100天计划！**

-END-

Python 技术

关于 Python 都在这里