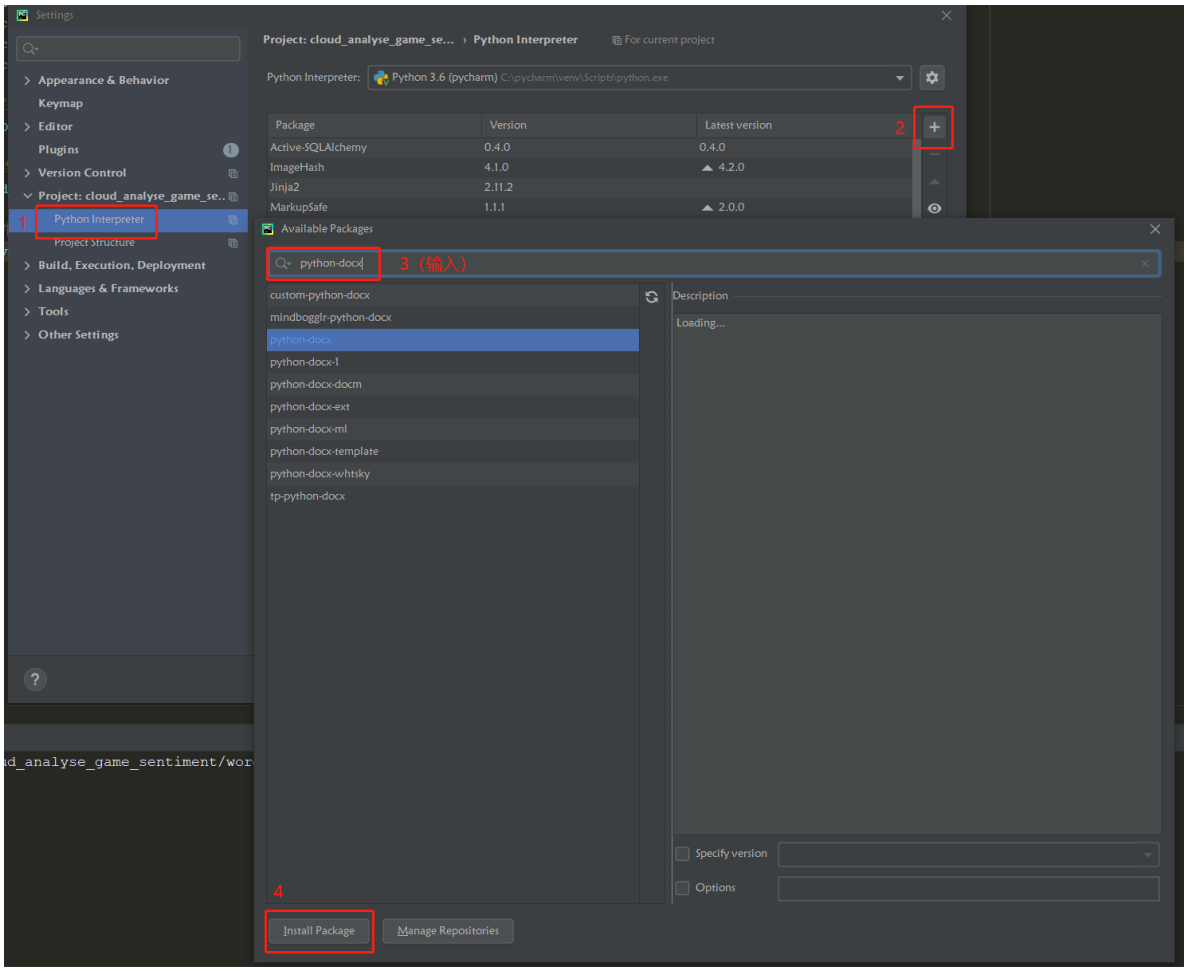
Task 03 python与word

3.0 课前准备

> python 处理 Word 需要用到 python-docx 库，终端执行如下安装命令：

```
pip3 install python-docx
或
conda install python-docx
```

> 或在pycharm的setting操作安装（示意如下）：



3.1.知识要点

项目难度：☆

> 说明：
>
> 1. 通过小试牛刀初步认识docx，然后系统学习python对word的操作；
> 2. 预估每个知识点需要讲解的时间；
> 3. 研发逻辑就是讲解逻辑，一般从上往下，遵循：`What - Why - How` 或 `Why - What - How` 思路；

3.1.1 初步认识docx

相信同学们都进行过word的操作。话不多说，直接上python对word简单操作的代码，先有个直观的感觉，然后再系统学习！

```
# 导入库
from docx import Document

# 新建空白文档
doc_1 = Document()

# 添加标题（0相当于文章的题目，默认级别是1，级别范围为0-9）
doc_1.add_heading('新建空白文档标题，级别为0',level = 0)
doc_1.add_heading('新建空白文档标题，级别为1',level = 1)
```

```
doc_1.add_heading('新建空白文档标题，级别为2',level = 2)

# 新增段落
paragraph_1 = doc_1.add_paragraph('这是第一段文字的开始\n请多多关照！')
# 加粗
paragraph_1.add_run('加粗字体').bold = True
paragraph_1.add_run('普通字体')
# 斜体
paragraph_1.add_run('斜体字体').italic =True

# 新段落（当前段落的下方）
paragraph_2 = doc_1.add_paragraph('新起的第二段文字。')

# 新段落（指定端的上方）
prior_paragraph = paragraph_1.insert_paragraph_before('在第一段文字前插入的段落')

# 添加分页符(可以进行灵活的排版）
doc_1.add_page_break()
# 新段落（指定端的上方）
paragraph_3 = doc_1.add_paragraph('这是第二页第一段文字！')

# 保存文件（当前目录下）
doc_1.save('doc_1.docx')
```

上节只是小试牛刀一下，接下来我们系统地学习python自动化之word操作。

在操作之前，我们需要了解 Word 文档的页面结构：

- 文档 - Document
- 段落 - Paragraph
- 文字块 - Run

`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。

在 `python-docx` 中，`run` 是最基本的单位，每个 `run` 对象内的文本样式都是一致的，也就是说，在从 `docx` 文件生成文档对象时，`python-docx` 会根据样式的变化来将文本切分为一个个的 `Run` 对象。

3.1.2 整体页面结构介绍

我们以一个小案例为主线把文档，段落和文字块串一下：

```
# 导入库
from docx import Document
from docx.shared import RGBColor, Pt,Inches,Cm
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT
from docx.oxml.ns import qn

# 新建文档（Datawhale）
doc_1 = Document()

# 字体设置（全局）
'''只更改font.name是不够的，还需要调用._element.rPr.rFonts的set()方法。'''

doc_1.styles['Normal'].font.name = u'宋体'
```

```
doc_1.styles['Normal']._element.rPr.rFonts.set(qn('w:eastAsia'), u'宋体')

# 添加标题（0相当于文章的题目，默认级别是1，级别范围为0-9，0时候自动带下划线）
heading_1 = doc_1.add_heading('周杰伦',level = 0)
heading_1.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER    #居中对齐，默认居左对齐

# 新增段落
paragraph_1 = doc_1.add_paragraph()
'''
设置段落格式：首行缩进0.75cm，居左，段后距离1.0英寸,1.5倍行距。
'''
paragraph_1.paragraph_format.first_line_indent = Cm(0.75)
paragraph_1.paragraph_format.alignment =  WD_PARAGRAPH_ALIGNMENT.LEFT
paragraph_1.paragraph_format.space_after =  Inches(1.0)
paragraph_1.paragraph_format.line_spacing =  1.5

text = '中国台湾华语流行歌手、' \
       '音乐创作家、作曲家、作词人、' \
       '制作人、杰威尔音乐公司老板之一、导演。' \
       '近年涉足电影行业。周杰伦是2000年后亚洲流行乐坛最具革命性与指标' \
       '性的创作歌手，有“亚洲流行天王”之称。他突破原有亚洲音乐的主题、形' \
       '式，融合多元的音乐素材，创造出多变的歌曲风格，尤以融合中西式曲风的嘻哈' \
       '或节奏蓝调最为著名，可说是开创华语流行音乐“中国风”的先声。周杰伦的' \
       '出现打破了亚洲流行乐坛长年停滞不前的局面，为亚洲流行乐坛翻开了新的一页！'

r_1 = paragraph_1.add_run(text)
r_1.font.size =Pt(10)     #字号
r_1.font.bold =True        #加粗
r_1.font.color.rgb =RGBColor(255,0,0)       #颜色

# 保存文件（当前目录下）
doc_1.save('周杰伦.docx')
```

通过上例我们可以看到，最小的操作对象为文字块，通过run的指定进行操作。比如字号，颜色等；而再上一个层级--段落是的格式是通过paragraph_format进行设置；

3.1.2字体设置

通过（1），同学们已经注意到，字体的设置是全局变量。如果我想在不同的部分进行不同字体的设置，那该怎么办呢？这就需要在应用前操作设置一下。

```
'''字体设置1.py'''
#导入库
from docx import Document
from docx.oxml.ns import qn
from docx.enum.style import WD_STYLE_TYPE

document = Document() # 新建docx文档

# 设置宋体字样式
style_font = document.styles.add_style('宋体', WD_STYLE_TYPE.CHARACTER)
style_font.font.name = '宋体'
document.styles['宋体']._element.rPr.rFonts.set(qn('w:eastAsia'), u'宋体')

# 设置楷体字样式
style_font = document.styles.add_style('楷体', WD_STYLE_TYPE.CHARACTER)
style_font.font.name = '楷体'
```

```
document.styles['楷体']._element.rPr.rFonts.set(qn('w:eastAsia'), u'楷体') # 将段落中的所有字体

# 设置华文中宋字样式
style_font = document.styles.add_style('华文中宋', WD_STYLE_TYPE.CHARACTER)
style_font.font.name = '华文中宋'
document.styles['华文中宋']._element.rPr.rFonts.set(qn('w:eastAsia'), u'华文中宋')

paragraph1 = document.add_paragraph() # 添加段落
run = paragraph1.add_run(u'aBCDefg这是中文', style='宋体') # 设置宋体样式

font = run.font #设置字体
font.name = 'Cambira' # 设置西文字体
paragraph1.add_run(u'aBCDefg这是中文', style='楷体').font.name = 'Cambira'
paragraph1.add_run(u'aBCDefg这是中文', style='华文中宋').font.name = 'Cambira'

document.save('字体设置1.docx')
```

```
'''字体设置2.py'''
#导入库
from docx import Document
from docx.oxml.ns import qn
from docx.enum.style import WD_STYLE_TYPE

#定义字体设置函数
def font_setting(doc,text,font_cn):
        style_add = doc.styles.add_style(font_cn, WD_STYLE_TYPE.CHARACTER)
        style_add.font.name = font_cn
        doc.styles[font_cn]._element.rPr.rFonts.set(qn('w:eastAsia'), font_cn)
        par = doc.add_paragraph()
        text = par.add_run(text, style=font_cn)

doc = Document()
a = '小朋友 你是否有很多问号'
b = '为什么 别人在那看漫画'
c = '我却在学画画 对着钢琴说话'

font_setting(doc,a,'宋体')
font_setting(doc,b,'华文中宋')
font_setting(doc,c,'黑体')

doc.save('字体设置2.docx')
```

我们很容易地看出来，字体设置1.py与字体设置2.py的区别在于是否为同一段落，同时字体设置2.py中自定义了一个函数。同学们可以在实际工作中看具体场景进行选择。

3.1.3插入图片与表格

```
#导入库
from docx import Document
from docx.shared import Inches

#打开文档
doc_1 = Document('周杰伦.docx')    #上面脚本存储的文档
#新增图片
```

```
doc_1.add_picture('周杰伦.jpg',width=Inches(1.0), height=Inches(1.0))

# 创建3行1列表格
table1 = doc_1.add_table(rows=2, cols=1)
table1.style='Medium Grid 1 Accent 1'  #表格样式很多种，如，Light Shading Accent 1等

# 修改第2行第3列单元格的内容为营口
table1.cell(0, 0).text = '营口'
# 修改第3行第4列单元格的内容为人民
table1.rows[1].cells[0].text = '人民'

# 在表格底部新增一行
row_cells = table1.add_row().cells
# 新增行的第一列添加内容
row_cells[0].text = '加油'

doc_1.save('周杰伦为营口加油.docx')
```

3.1.4设置页眉页脚

在python-docx包中则要使用节(section)中的页眉(header)和页脚(footer)对象来具体设置。

```
from docx import Document
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT

document = Document() # 新建文档

header = document.sections[0].header # 获取第一个节的页眉
print('页眉中默认段落数：', len(header.paragraphs))
paragraph = header.paragraphs[0] # 获取页眉的第一个段落
paragraph.add_run('这是第一节的页眉') # 添加页面内容
footer = document.sections[0].footer # 获取第一个节的页脚
paragraph = footer.paragraphs[0] # 获取页脚的第一个段落
paragraph.add_run('这是第一节的页脚') # 添加页脚内容


'''在docx文档中又添加了2个节，共计3个节，页面和页脚会显示了“与上一节相同”。
如果不使用上一节的内容和样式要将header.is_linked_to_previous的属性或
footer.is_linked_to_previous的属性设置为False，
用于解除“链接上一节页眉”或者“链接上一节页脚”。'''
document.add_section() # 添加一个新的节
document.add_section() # 添加第3个节
header = document.sections[1].header # 获取第2个节的页眉
header.is_linked_to_previous = False # 不使用上节内容和样式

#对齐设置
header = document.sections[1].header # 获取第2个节的页眉
header.is_linked_to_previous = False # 不使用上节内容和样式
paragraph = header.paragraphs[0]
paragraph.add_run('这是第二节的页眉')
paragraph.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER # 设置页眉居中对齐
document.sections[1].footer.is_linked_to_previous = False
footer.paragraphs[0].add_run('这是第二节的页脚') # 添加第2节页脚内容
footer.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.CENTER # 设置第2节页脚居中
对齐
header = document.sections[2].header # 获取第3个节的页眉
```

```
header.is_linked_to_previous = False # 不使用上节的内容和样式
paragraph = header.paragraphs[0] # 获取页眉中的段落
paragraph.add_run('这是第三节的页眉')
paragraph.alignment = WD_PARAGRAPH_ALIGNMENT.RIGHT # 设置页眉右对齐
document.sections[2].footer.is_linked_to_previous = False
footer.paragraphs[0].add_run('这是第三节的页脚') # 添加第3节页脚内容
footer.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.RIGHT # 设置第3节页脚右对齐
document.save('页眉页脚1.docx') # 保存文档
```

结果如下:

3.1.5代码延伸

```
'''对齐设置'''
from docx.enum.text import WD_ALIGN_PARAGRAPH
#LEFT: 左对齐
#CENTER: 文字居中
#RIGHT: 右对齐
#JUSTIFY: 文本两端对齐

'''设置段落行距'''
from docx.shared import Length
# SINGLE :单倍行距（默认）
#ONE_POINT_FIVE : 1.5倍行距
# DOUBLE2 : 倍行距
#AT_LEAST : 最小值
#EXACTLY:固定值
# MULTIPLE : 多倍行距

paragraph.line_spacing_rule = WD_LINE_SPACING.EXACTLY #固定值
paragraph_format.line_spacing = Pt(18) # 固定值18磅
paragraph.line_spacing_rule = WD_LINE_SPACING.MULTIPLE #多倍行距
paragraph_format.line_spacing = 1.75 # 1.75倍行间距

'''设置字体属性'''
from docx.shared import RGBColor,Pt
#all_caps:全部大写字母
#bold:加粗
#color:字体颜色
```

```
#double_strike:双删除线
#hidden : 隐藏
#imprint : 印记
#italic : 斜体
#name  :字体
#shadow  :阴影
#strike  :  删除线
#subscript  :下标
#superscript  :上标
#underline  :下划线
```

3.2 项目实践

项目难度：☆ ☆ ☆

3.2.1需求

> 你是公司的行政人员，对合作伙伴进行邀请，参加公司的会议；
>
> 参会人名单如下：

	A	B	C	D
1	公司	职位	姓名	邀请时间
2	阿里	数据工程师	牛云	2021/2/15
3	腾讯	数据分析师	牛化腾	2021/2/16
4	百度	数据架构师	张艳红	2021/2/17
5	京东	算法工程师	王强东	2021/2/18
6				

拟定的邀请函样式如下：

-

邀 请 函

尊敬的京东公司算法工程师王强东，您好：

现诚挚的邀请您于 2021 年 10 月 27 日参加 DataWhale 主办的享受开源 2050 活动，地点在北京鸟巢，希望您届时莅临参加。

邀请时间：2021 年 02 月 18 日

根据参会人名单，利用python批量生成邀请函。

3.2.2需求分析

> 逻辑相对简单：
>
> - 获取 Excel 文件中每一行的信息，提取 参数；结合获取的参数设计邀请函样式并输出
> - 设计word段落及字体等样式。

3.2.3代码

```
# 导入库
from openpyxl import load_workbook
from docx import Document
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT
from docx.shared import RGBColor, Pt,Inches,Cm
from docx.oxml.ns import qn


path = r'D:\idea\cloud_analyse_game_sentiment\word自动化'
# 路径为Excel 文件所在的位置，可按实际情况更改

workbook = load_workbook(path + r'\excel到word.xlsx')
sheet = workbook.active    #默认的WorkSheet

n = 0    #为了不遍历标题（excel的第一行）
for row in sheet.rows:
    if n:
        company = row[0].value
        office = row[1].value
        name = row[2].value
        date = str(row[3].value).split()[0]
        print(company, office, name, date)


        doc = Document()
        heading_1 = '邀 请 函'
        paragraph_1 = doc.add_heading(heading_1, level=1)
        # 居中对齐
        paragraph_1.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER
        # 单独修改较大字号
        for run in paragraph_1.runs:
            run.font.size = Pt(17)

        greeting_word_1 = '尊敬的'
        greeting_word_2 = '公司'
        greeting_word_3 = ', 您好：'
        paragraph_2 = doc.add_paragraph()

        paragraph_2.add_run(greeting_word_1)
        r_1 = paragraph_2.add_run(company)
        r_1.font.bold = True  # 加粗
        r_1.font.underline = True    #下划线

        paragraph_2.add_run(greeting_word_2)

        r_2 = paragraph_2.add_run(office)
        r_2.font.bold = True  # 加粗
        r_2.font.underline = True    #下划线

        r_3 = paragraph_2.add_run(name)
        r_3.font.bold = True  # 加粗
        r_3.font.underline = True    #下划线
        paragraph_2.add_run(greeting_word_3)

        paragraph_3 = doc.add_paragraph()
```

```
        paragraph_3.add_run('现诚挚的邀请您于2021年10月27日参加Datawhale主办的享受开源2050活动，地点在北京鸟巢，希望您届时莅临参加。')
        paragraph_3.paragraph_format.first_line_indent = Cm(0.75)
        paragraph_3.paragraph_format.alignment = WD_PARAGRAPH_ALIGNMENT.LEFT
        paragraph_3.paragraph_format.space_after = Inches(1.0)
        paragraph_3.paragraph_format.line_spacing = 1.5

        paragraph_4 = doc.add_paragraph()
        date_word_1 = '邀请时间：'
        paragraph_4.add_run(date_word_1)
        paragraph_4.alignment = WD_PARAGRAPH_ALIGNMENT.RIGHT
        sign_date = "{}年{}月{}日".format(date.split('-')[0], date.split('-')[1], date.split('-')[2])
        paragraph_4.add_run(sign_date).underline = True
        paragraph_4.alignment = WD_PARAGRAPH_ALIGNMENT.RIGHT

        #设置全文字体
        for paragraph in doc.paragraphs:
            for run in paragraph.runs:
                run.font.color.rgb = RGBColor(0, 0, 0)
                run.font.name = '楷体'
                r = run._element.rPr.rFonts
                r.set(qn('w:eastAsia'), '楷体')
        doc.save(path + "\{}-邀请函.docx".format(name))
    n = n + 1
```

3.3 后记

> 本案例也可适用于批量生产固定格式的word，如工资条，通知单等，面对这种相似且重复的任务，python的自动化运行能大幅提升当前的工作效率。