高效办公O2复习资料

恭喜oa高效办公全部课程已经完成啦~

完课不是终点，而是新的开始，希望大家能好好利用这些天里学习的知识点，好好想想如何解决自己工作中的“重复工作”噢。

——第6关——

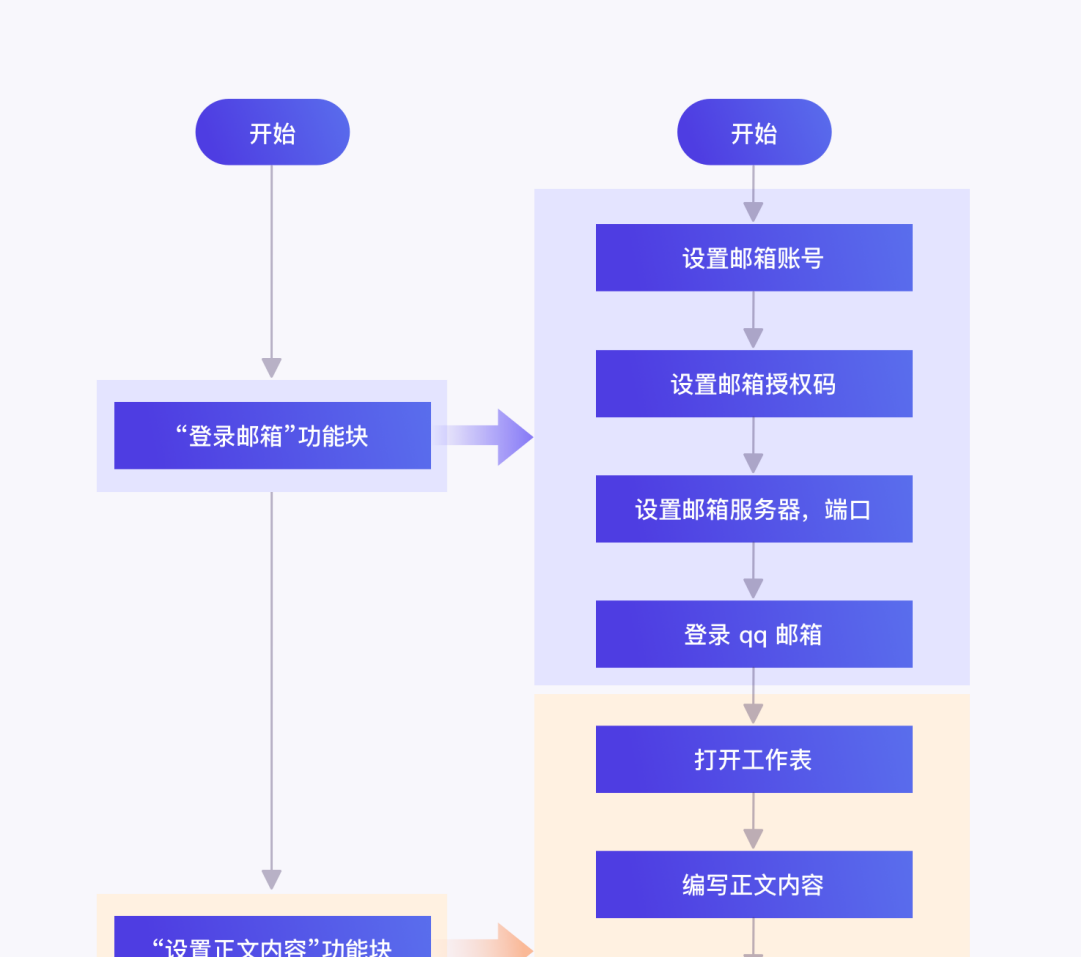
这一关我们学习了使用 python 批量发送 email 邮件。

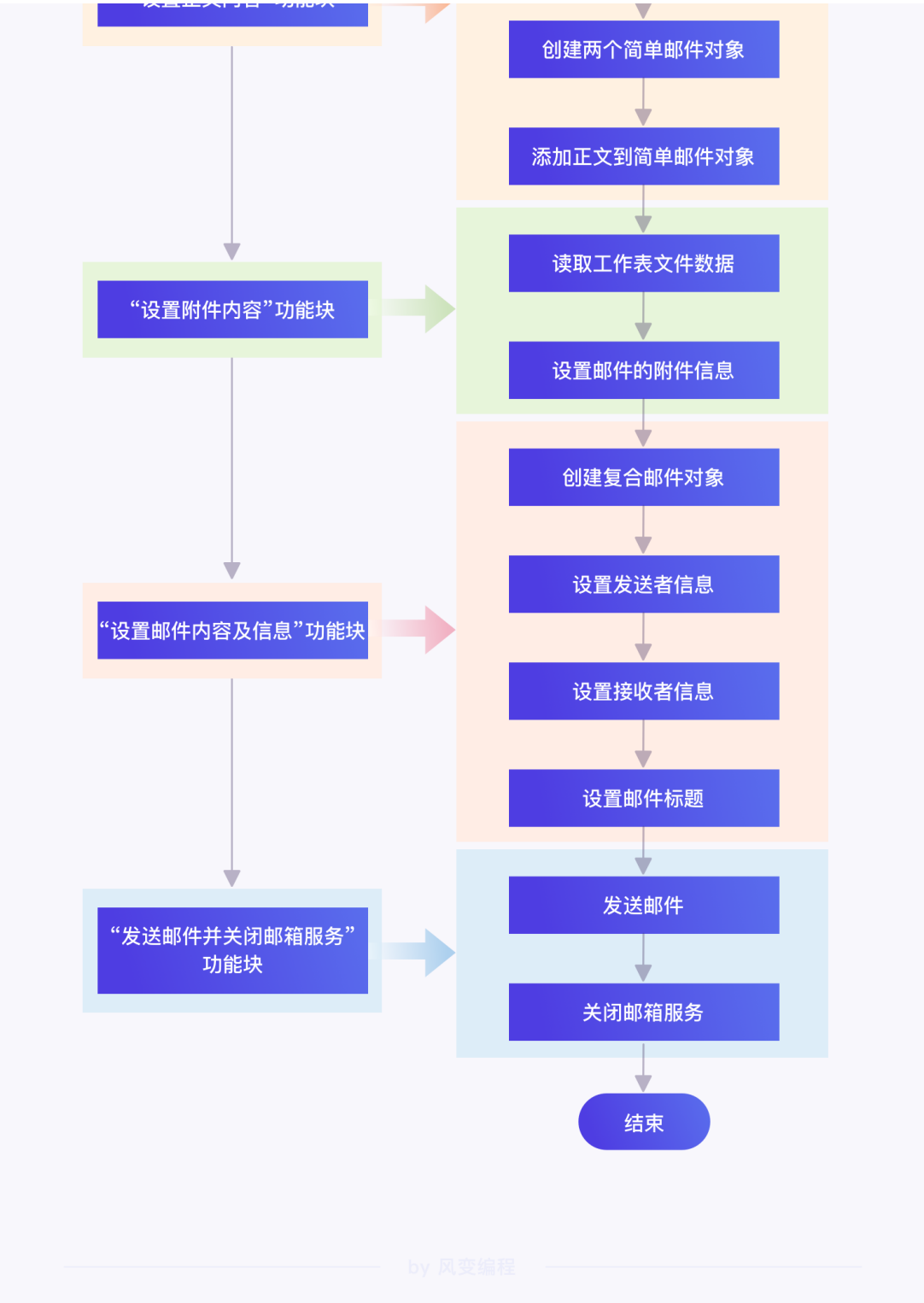
其实用代码发邮件和我们手工发电子邮件的步骤一样，主要分为三步：

1. 登录邮箱
2. 新建并编辑邮件
3. 发送邮件并关闭邮箱服务

其中，新建并编辑邮件又包括这三个步骤：① 设置正文内容；② 设置附件内容；③ 设置邮件内容及信息。

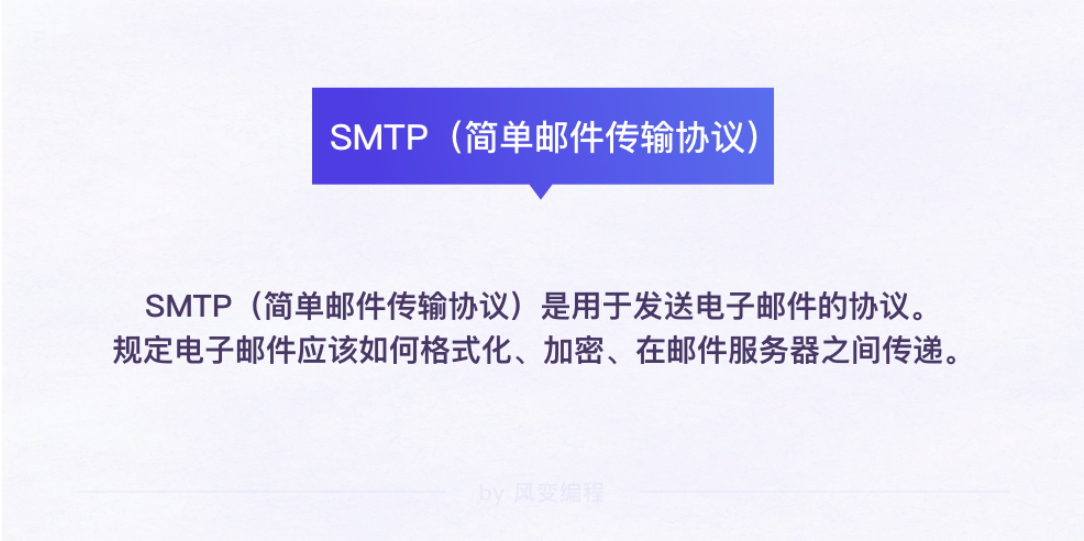
主要看看下面这张流程图：





主要用到的是 smtplib 模块和 email 模块。

smtplib 是 Python 的内置模块，它对 SMTP 协议进行了简单的封装，提供了一种发送电子邮件的方法。



登录邮箱分为两个步骤。

第一步：我们需要实例化模块 smtplib 中的类 SMTP_SSL()，得到一个 smtp 对象。

```
smtp = smtplib.SMTP_SSL(host, port)
```

这里我们需要传入两个参数：参数1：host（服务器地址），参数2：port（端口号）。

服务器地址，一般是公开的一个网站地址，相当于是该服务在互联网中的标识。
而端口号就好比一个通道，通向这个网站特定的一个服务。

不同的邮箱服务器地址和端口号都是不同的，可以通过百度查询得到。

第二步是登录qq邮箱账号，这里需要用到方法login()。

```
smtp.login('example@mail.com',授权码)
```

这里我们要传入两个参数：1.邮箱地址、2. 邮箱授权码。

需要注意的是这里用的是邮箱授权码，与登录密码不同，授权码是用于登录第三方客户端的专用密码，目的是防止你的邮箱密码泄露，造成信息泄露。

登录了邮箱之后，下一步是设置正文内容。

编辑正文文本，我们需要实例化类 MIMEText()，得到一个简单邮件对象。 而简单邮件对象可以用来承载邮件内容，例如：正文文本、附件。

```
email_content = MIMEText(_text, _subtype, _charset)
```

实例化MIMEText()时，需要往括号内传入三个参数：参数1：_text 、参数2：_subtype、参数3：_charset 。

参数1：_text，意思就是内容。例如：构造正文时就是正文文本，构造附件时就是读取到的附件文件。

参数2：_subtype，意思是文本格式。构造正文内容时文本格式为'plain'，而构造附件时文本格式为'base64'。

参数3：_charset，意思是编码方式。此处采用'utf-8'，它是电子邮件、网页及其他存储或传送文字的应用中，常常采用的编码。

参数：_text	参数：_subtype	参数：_charset
正文文本	plain	utf-8
附件	base64	utf-8

而如果我们要设置附件则需要分为三步：

第一步是读取附件内容，即代码中的前三行代码，将读取到的内容赋给file_data。

```
 # 读取工作表文件数据
with open('./04_月考勤表.xlsx', 'rb') as f:
    file_data = f.read()
```

第二步是设置内容类型为附件。

```
# 设置内容类型为附件
attachment = MIMEText(file_data, 'base64', 'utf-8')
```

最后一步是设置附件标题及文件类型，使其能以邮件中附件的形式发送出去。
这里我们需要用到的方法是：add_header()：

```
# 设置附件标题以及文件类型
attachment.add_header('Content-Disposition', 'attachment', filename='04_月考勤表.xlsx')
```

插入附件时，前面俩参数固定为：'Content-Disposition','attachment'，其中filename是可以自己命名的，写什么名字，邮件中附件就显示什么名字。

而当我们邮件内容包括正文和附件的两个内容的时候，就需要实例化复合邮件对象。它可以用来装载多个简单邮件对象。

而构建复合邮件内容时，要用到email模块下mime模块下模块中的类：MIMEMultipart()。
和简单邮件对象MIMEText()一样，我们需要对其实例化：

```
# 实例化复合邮件对象
msg = MIMEMultipart()
```

简单邮件对象(MIMEText对象)，像一个物件，可以是一段文本，一个附件，可以单独作为邮件内容发出。

复合邮件对象(MIMEMultipart对象)，像一个容器，由多个简单邮件对象组合而成，一起作为邮件内容发出。

将简单邮件对象添加到复合邮件对象msg中需要用到 attach() 方法。 其语法格式为：msg.attach(简单邮件对象)

```
# 创建复合邮件对象
msg = MIMEMultipart()
# 添加正文到复合邮件对象中
msg.attach(email_content)
# 添加附件到复合邮件对象中
msg.attach(attachment)
```

上面的代码也就是将编辑好的正文：email_content，附件：attachment，用 attach() 方法加入到复合邮件对象中的过程。

完成邮件内容编辑之后，我们还需要设置邮件的信息，比如收件人发件人标题等，语法是这样的：

```
# 实例化复合邮件对象
msg = MIMEMultipart()
# 设置发送者信息
msg['From'] = '陈知枫'
# 设置接受者信息。
msg['To'] = '闪光科技的同事们'
# 设置邮件标题
msg['Subject'] = '04_月考勤表'
```

其中，msg['From']用于设置发送者信息、msg['To']用于设置接受者信息、msg['Subject']用于设置文件标题 ，类似于字典给键赋值。

完成了上述对邮件的设置，接下来就是发送邮件以及关闭邮箱服务，需要用到smtplib模块里面的两个方法：sendmail()、quit()。

首先是sendmail()，它是smtplib模块中发送邮件的方法，下面是它的语法格式：

```
# 发送邮件
smtp.sendmail(from_addr, to_addrs, msg.as_string())
```

需要传入三个参数：参数1：from_addr（发件邮箱地址）、参数2：to_addrs（收件邮箱地址）、参数3：msg.as_string()（邮件内容）。

msg 是需要发送的邮件内容对象，可以是简单邮件对象也可以是复合邮件对象。

as_string() 是将发送的信息 msg 变为字符串类型，记得在传输邮件时你也要用 as_string() 将邮件内容转化为字符串。

关闭邮件服务，需要用smtplib模块中的方法：quit() 。对smtp对象调用quit()方法，无需传入参数即可直接关闭邮箱服务：

```
# 关闭邮箱服务
smtp.quit()
```

——第7关——

第7关我们主要学习了利用 python-docx 模块处理 word 文件。

python-docx 库是专为 Python 设计的，用于创建和处理 Microsoft Word (.docx) 文件的第三方库。

需要注意的是，安装时输入python-docx，但写代码导入这个库时，直接写docx即可 ：

```
# 安装 python-docx 库
pip install python-docx

# 导入 python-docx 库
import docx
from docx import ...
```

同时，python-docx 库只能处理 docx 文件（ Microsoft office 2007 之后的 office 版本生成的 Word 文档格式），它是无法处理 doc 文件的。

python-docx 库在处理 Word 文件时，一般会将这个文件分为三个层级：

- 最外层的文档对象（Document）；
- 中间的块级元素（Block-level）；
- 最内层的，包含在块级元素里面的内联元素（Inline-level）。

文档对象（Document）、块级元素（Block-level）和内联元素（Inline-level）这种由外层到内层的关系，可以理解为文档、段落和段落内的文字。

首先是 Document 对象，我们可以通过实例化 Document 类的方式，把一个 Word 文件读取为 python-docx 中的 Document 文档对象，以便对文档进行后续的操作。

一般情况下，我们会把类 Document 从 python-docx 库中直接导入，即使用from docx import Document导入 Document 类。 语法如下：

```
# 导入模块
from docx import Document

# 实例化Document对象
变量名 = Document(docx)
```

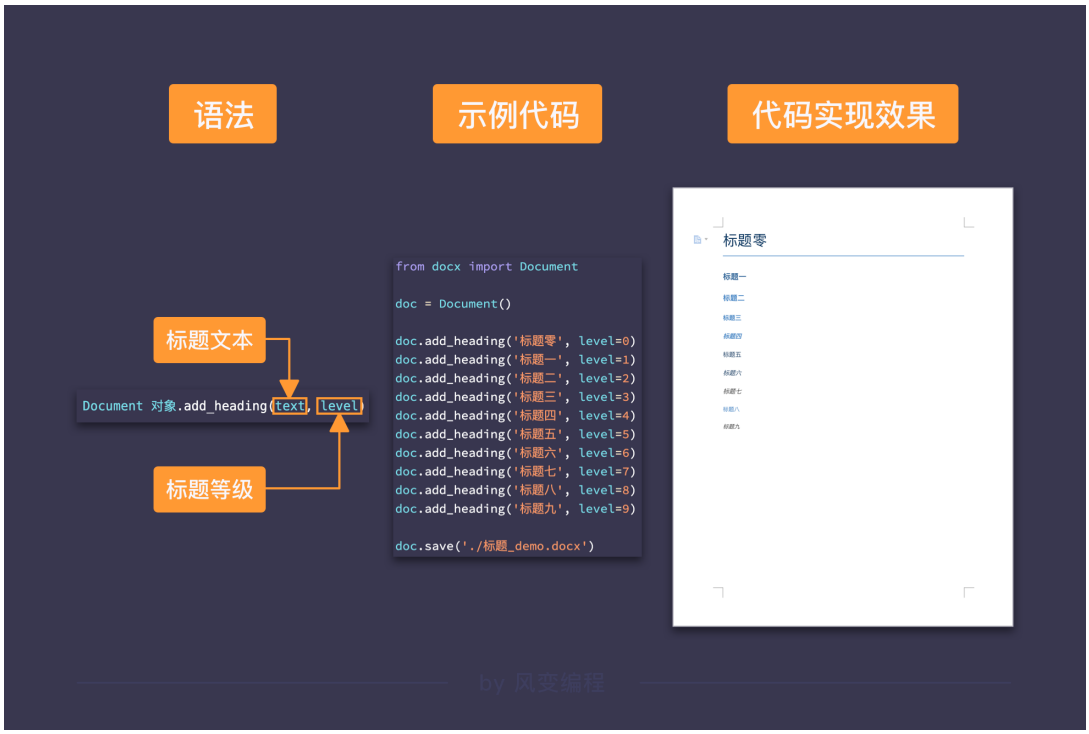
参数 docx 表示 Word 文件的路径。这里存在两种情况：

如果传入路径，会把路径代表的 Word 文件读取成一个 Document 对象。
如果不传入路径，就会生成一个空的 Document 对象，相当于新建了一个空白的 Word 文件。

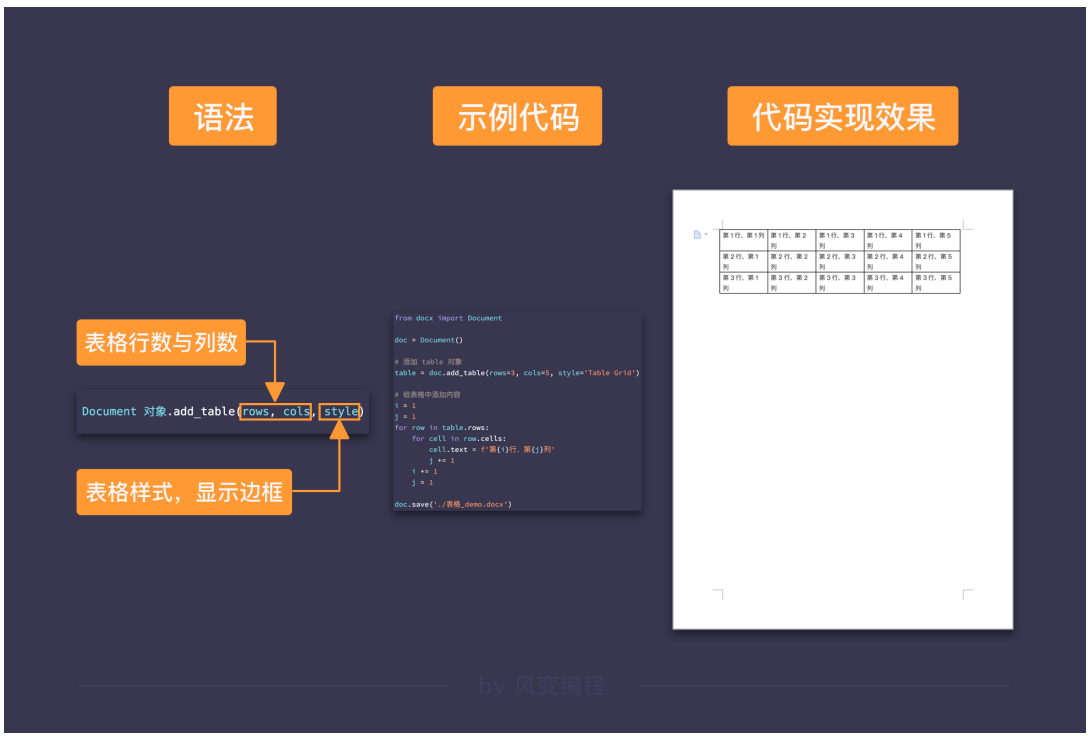
等所有的操作都结束后，如果我们想把对文档的操作结果保存起来，要使用如下语句保存文档。

```
Docuemnt对象.save(path)
```

然后我们讲到了添加标题（Heading 对象）的方法：Docuement 对象.add_heading()。



以及添加表格（Table 对象）的方法：Docuement 对象.add_table()。



这两个方法大家稍作了解即可，重点在于当我们需要对文档添加文字和图片的时候，需要用到的 Paragraph 对象 和 Run 对象。

在 Word 文件里，，每按一次回车，光标进入下一段，就意味着多了一个段落。每一个段落都对应着一个 Paragraph 对象。

我们可以通过 Document 对象的属性 paragraphs 来查看一个 word 文件中的所有段落。

Document 对象.paragraphs会返回一个列表，列表中的每一个元素都是一个 Paragraph 对象。还可以通过 Paragraph 对象的属性text来查看段落中的文字。

```
# 获取文档的所有段落对象并返回一个列表
变量名 = Document对象.paragraphs
```

给 Document 添加 Paragraph 对象则需要调用Document 对象的方法add_paragraph() 。

```
# 给 Document 添加 Paragraph 对象
Document对象.add_paragraph()
```

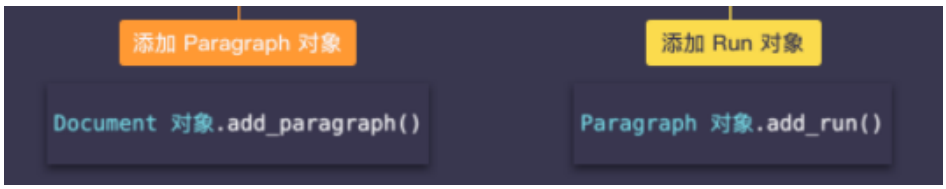
这个方法会在 Document 对象末尾添加一个 Paragraph 对象，即添加一个段落。空文档时，在一开始的空白处添加段落，但当段落中有内容时，在最后的内容后另起一行添加新的段落。

而Run 对象则是 Paragraph 对象中具有相似样式（如相似的字体大小，字体形状和字体样式）的文字，也可能是一张图片。

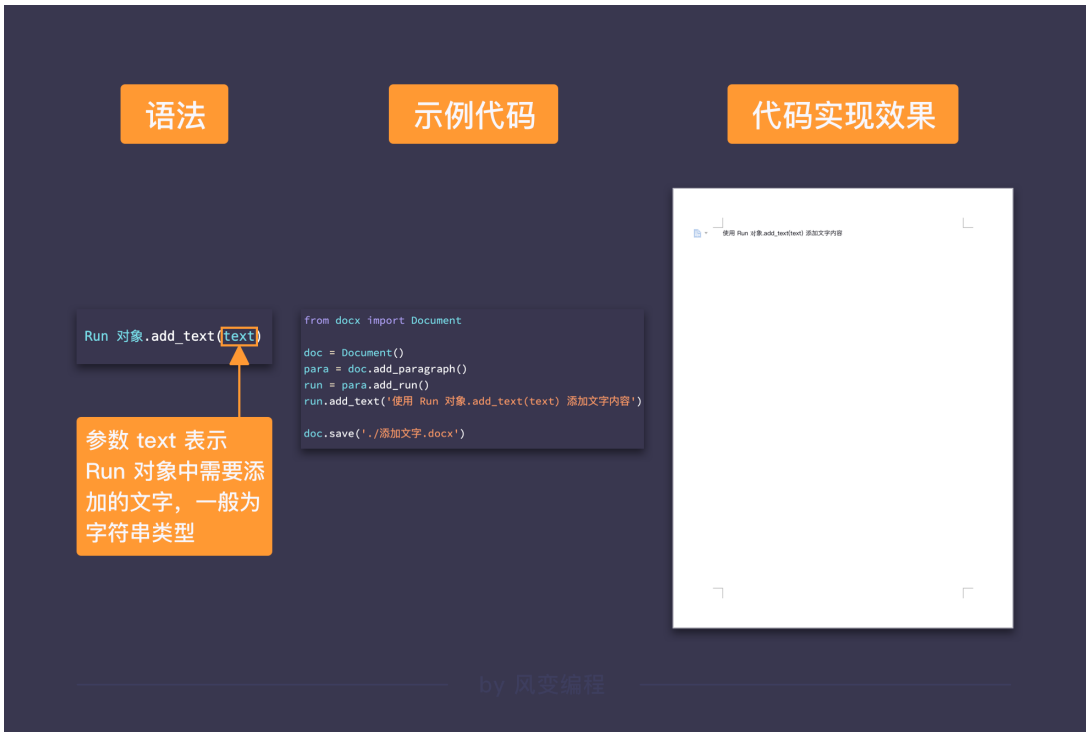
要查看Paragraph 对象中的 Run 对象可以调用 Paragraph 对象的属性 runs。

Paragraph 对象.runs 也会返回一个列表，列表中的每一个元素都是一个 Run 对象。
与 Paragraph 对象类似，Run 对象也有 text 属性，所以可以使用该属性查看其中的文字。

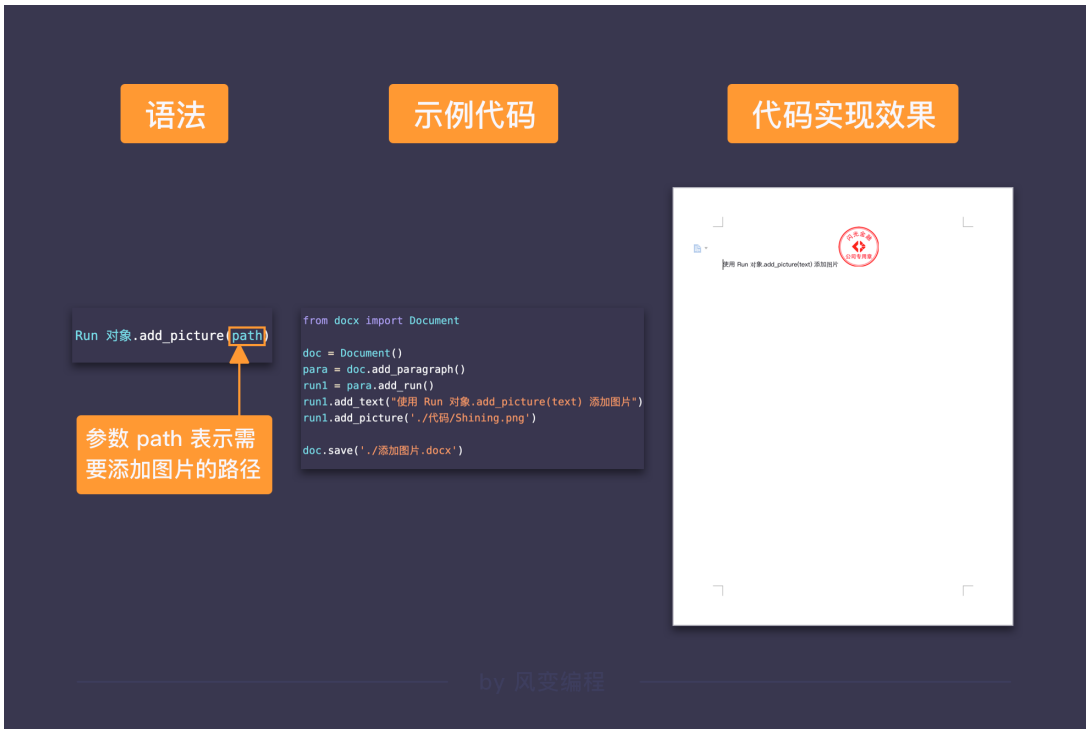
给 Paragraph 添加 Run 对象的方式跟前面类似，调用 Paragraph 对象的方法add_run()就可以了，即写为Paragraph 对象.add_run() 。



这个时候，添加文字需要 Run 对象调用方法add_text(text)，其语法及其效果如下图所示：



添加图片需要 Run 对象去调用add_picture(path)方法，语法和效果见下图：



学习完这些知识之后，我们还可以对添加的文字进行样式的设置。

可以使用 Paragraph 对象的属性paragraph_format 来控制段落的样式，即段落在页面的位置，例如对齐方式、缩进和段落前后的间距。

要设置对齐方式的话，可以这么写：paragraph_format.alignment

如果要具体指定段落是左对齐、右对齐、居中对齐还是两端对齐，就需要在一开始导入WD_ALIGN_PARAGRAPH，具体的写法是：

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

四种对齐方式的写法：

4 种对齐方式

from docx.enum.text import WD_ALIGN_PARAGRAPH	
左对齐（默认）	Paragraph 对象.paragraph_format.alignment = WD_ALIGN_PARAGRAPH.LEFT
右对齐	Paragraph 对象.paragraph_format.alignment = WD_ALIGN_PARAGRAPH.RIGHT
居中对齐	Paragraph 对象.paragraph_format.alignment = WD_ALIGN_PARAGRAPH.CENTER
两边对齐	Paragraph 对象.paragraph_format.alignment = WD_ALIGN_PARAGRAPH.JUSTIFY

by 风变编程

至于缩进、间距等样式可以自行查阅官方文档：https://python-docx.readthedocs.io/en/latest/user/text.html

除了这些，我们还可以使用 Run 对象的属性font来设置字体的样式。

当我们需要设置字号时，需要先导入Pt，即写为：

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

字号对应关系

中文字号	Pt 字号
初号字	42pt
小初号字	36pt
一号字	26pt
小一号字	24pt
二号字	22pt
小二号字	18pt
三号字	16pt
小三号字	15pt
四号字	14pt
小四号字	12pt
五号字	10.5pt
小五号字	9pt
六号字	7.5pt
小六号字	6.5pt

然后再让属性 font 调用 size，根据上面的表格，将导入的 Pt 设置好后赋值给 size 就可以设置字体的大小了：

```
from docx.shared import Pt
# 设置字体大小
Run 对象.font.size = Pt(14) # 字体大小
```

还可以通过属性 font 调用 bold，然后将值设置为 True 设置字体加粗：

```
# 设置字体加粗
Run 对象.font.bold = True # 字体加粗
```

font 属性的其他功能参考下图：

font 补充

from docx.shared import RGBColor, Pt		
参数	说明	示例
指定字体	Run 对象.font.name	font.name = 'Calibri'
设置斜体	Run 对象.font.italic	font.italic = True
设置下划线	Run 对象.font.underline	font.underline = True
设置字体颜色	Run 对象.font.color.rgb	font.color.rgb = RGBColor(255, 0, 0)

python-docx 的功能当然远远不止这些，我们肯定也没办法全部在课程里教给大家，但是大家掌握了学习的方法之后，就可以自己根据需要去学习啦~