PYTHON程序设计

计算机学院 王纯

八 文件操作

八 文件操作

- 文件和流
- 文件对象
- 文件的基本操作
- 随机文件读写
- 其他文件读写
- 对象序列化
- 常用办公文件格式的处理

文件和流

文件　是持久化的数据

流　可以读一次或有限次数的数据序列（流的起点和终点）

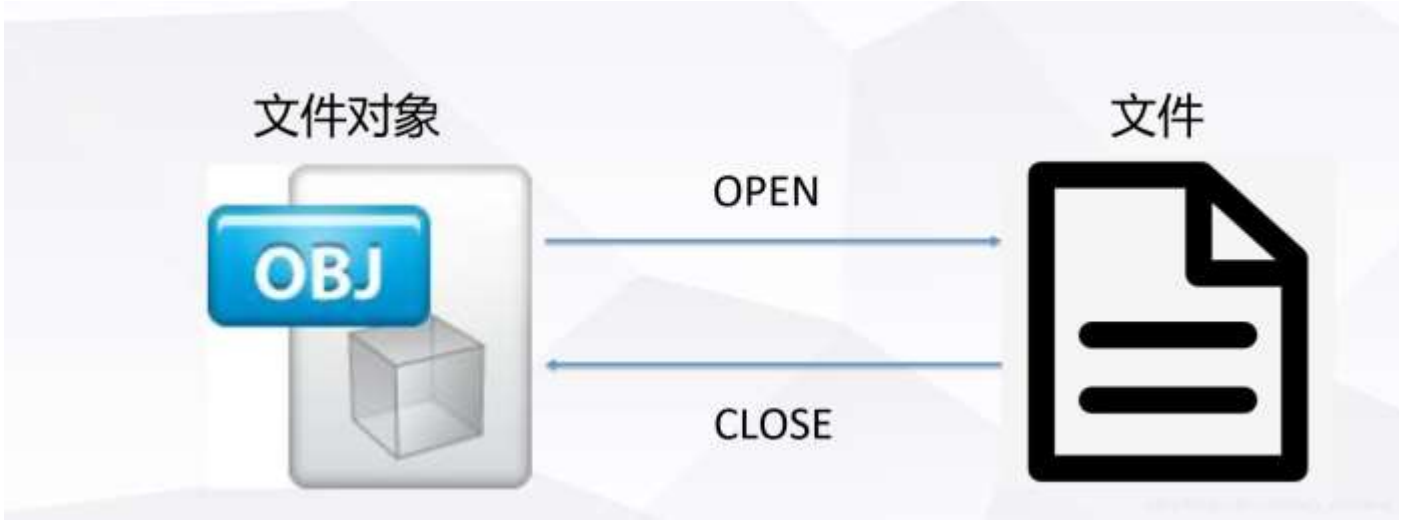
输入流和输出流　对一个流，立足点不同： 流的起点-输出流，流的终点-输入流

```
# demo
import os

try:
    file1 = open('dat', 'w') # 打开当前目录下的文本文件dat以便写入，file1为文件对象
        # 实际上dat也可以换成完整的路径+文件名
        file1.write('10000') # 写入字符串10000
except FileNotFoundError: # 文件读写操作可能发生异常，这里捕获三种异常，文件不存在、
        # 没有足够操作权限和其它异常
        print('File not found error\n')
except PermissionError:
    print('Permission error\n')
except BaseException:
    print('Other unexpected error')
finally:
    file1.close() # 不论文件操作正确与否，都必须关闭文件
```

文件对象

文件对象 实质上是文件在Python运行时的代理



文件对象

一种简化的文件操作方法：使用***with***语句

```
with open('dat','w') as f:
    f.write('100')
    f.write('300')
```

with ... as ... 的用法

with <expression> as target：

- ✓其中expression是上下文管理器表达式，target是资源对象
- ✓这种用法适用于对资源进行访问的场合，确保不管使用过程中是否发生异常都会执行必要的“清理”操作，释放资源，with 语句主要是为了简化代码
- ✓上下文管理器指实现了__enter__和__exit__的类（即满足上下文协议）

文件对象

打开文件

f1=open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)

关闭文件

f1.close() 文件修改被保存并且释放文件对象资源。

文件打开方式

mode	说明
r	表示只读方式打开文本文件， 可从文件读出数据， 但不能写入数据。
w	表示只写方式打开文件， 如果该文件已经存在， 则以空文件覆盖存在的文件。
x	表示独占创建文件， 如果文件已经存在， 则以此模式打开文件就会出现失败。
a	表示打开文件写， 不清空文件， 在文件后尾以追加的方式写入。
b	表示二进制模式打开文件。 该模式可以和前面的这些模式合在一起用。 例如ab，xb等等。
t	表示文本模式打开文件， 是默认值， 可省略。
+	表示以更新方式打开磁盘文件， 可读可写

文件的基本操作：文本文件的写入

打开文件时mode参数的设置：

‘*w*’ ：只写方式

‘*a+*’ ， ‘*r+*’ ， ‘*w+*’ ：读写方式

f为文件对象：

f.write(s)：把字符串s写入文件中

f.writelines(lines)：把lines中的多个字节序列写入文件

f.flush()：强制立即将缓冲区中数据写入到文件中

```
# demo0802.py 文件写入
BasePath = 'c:\\data\\'
try:
    f1 = open(BasePath + 'a1.txt', 'w')  # 若a1.txt存在，则清空其内容再写入
    f2 = open(BasePath + 'b1.txt', 'r+')  # b1.txt必须存在。r+可支持写入
    f3 = open(BasePath + 'c1.txt', 'w+')  # c1如果不存在，则自动创建
    f4 = open(BasePath + 'd1.txt', 'a+')  # 添加模式，在文件末尾添加数据
    f1.write('123')  # 清空原有内容，重新写入新的数据
    # f1.write(123) #错误，只能写入字符串
    f2.write('123')  # 只覆盖当前位置（起始位置）的3个字符的内容，但不影响后面的内容
    print("b1:", f2.read())  # 读出的是b1.txt的第4个字符到文件末尾的全部内容
    f3.write('123')  # 在当前文件位置（起始位置）写入新数据
    print("c1:", f3.read())  # 读出内容为空
    f4.write('abc')  # 在文件末尾添加新的内容
    print("d1:", f4.read())  # 读出内容为空
except IOError as e:
    print(e)
    exit()
finally:
    f1.close()
    f2.close()
    f3.close()
    f4.close()
```

文件的基本操作：文本文件的读出

打开文件时mode参数的设置：

'*r*'：读方式

'*a+*'，'*r+*'，'*w+*'：为读写方式

f为文件对象：

f.read()：读取f当前指针位置到文件末尾的全部内容，文件指针推进到文件末尾，返回一个字符串

f.read(n)：读取n个字符，返回一个字符串

f.readline()：读取一行，返回一个字符串

f.readlines()：读取多行，返回一个字符串列表

```
# demo0803.py 文件读出
BasePath = 'c:\\data\\'
try:
    f1 = open(BasePath + 'a11.txt', 'r')  # a11.txt必须存在，否则出错
    f2 = open(BasePath + 'b11.txt', 'r+')  # b11.txt必须存在。r+可读可写。
    # f1.write('123') #错误：只能读不能写
    print("f1.read(3):", f1.read(3))  # 从文件中读取3个字符
    print("f1.readline():", f1.readline())  # 从文件中读取一行内容
    print("f1.read():", f1.read())  # 读取文件的全部剩余内容
    f2.write('123')  # 覆盖当前位置3个字符的内容，但不影响后面的内容
    print("f2.read():", f2.read())  # 读出的是b11.txt的第4个字符到文件末尾的全部内容
except IOError as e:
    print(e)
    exit()
finally:
    f1.close()
    f2.close()
```

文件的基本操作：二进制文件的写入

打开文件时mode参数的设置：

‘*wb*’ ：只写方式

‘*ab+*’ ， ‘*rb+*’ ， ‘*wb+*’ ：读写方式

f为文件对象：

f.write(b)：将字节数据b写入f对应的打开文件 中。返回实际写入字节数

```
# demo0804.py 二进制文件写入
BasePath = 'c:\\data\\'
try:
    f1 = open(BasePath + 'a21.txt', 'wb’)
    # f1.write('123') #错误：不能写入字符串，只能写入字节数据
       f1.write(b'123abc')
    a = bytearray(32)
    for ch in range(0, 32):
        a[ch] = 65  # 65为ASCII字符A
    f1.write(a)
    for ch in range(0, 32):
        a[ch] = ch  # ASCII字符0-31
    f1.write(a)
    # print(f1.read(3)) #不可从文件中读
except IOError as e:
    print("error:", e)
    exit()
finally:
    f1.close()
```

文件的基本操作：二进制文件的读出

打开文件时mode参数的设置：

‘*rb*’ ：读方式

‘*ab+*’ ， ‘*rb+*’ ， ‘*wb+*’ ：为读写方式

f为文件对象：

f.read()：从f中读取剩余内容直到文件结尾，返 回一个bytes对象

f.read(n)：从f中读取n个字节，返回一个bytes 对象，如果n为负数或None，则等同于 *f.read()*

f.readinto(b)：从f中读取*len(b)*个字节写入bytes 对象b，如果文件剩余内容长度不到b的长度， 则读出全部内容。返回值为实际读出的长度

```
# demo0805.py 二进制文件读出
BasePath = 'c:\\data\\'
try:
    f1 = open(BasePath + 'a21.txt', 'rb')  # a21.txt必须存在，否则出错
        # f1.write('123') #错误：只能读不能写
        print(f1.read(3))  # 从文件中读取3个字符
        b = bytearray(10)
    print("f1.readinto(b):", f1.readinto(b))  # 从文件中读取10个字节
        print("b:", b)
    # f1.seek(0) #表示将文件位置指针移动到文件开始位置
        print("f1.read():", f1.read())  # 读取文件的全部剩余内容
        print("f1.readinto(b):", f1.readinto(b))  # 因为已经到文件尾，所以读出0个字节
except IOError as e:
    print(e)
    exit()
finally:
    f1.close()
```

随机文件读写

Python并不区分顺序文件还是随机文件

打开方法和顺序文件相同

设置读写指针位置

f.seek(offset,whence=os.SEEK_SET | os.SEEK_END| os.SEEK_CUR)

#f为文件对象

通常对二进制文件随机读写

```
# demo0806.py
import os

BasePath = 'c:\\data\\'
try:
    f1 = open(BasePath + 'a21.txt', 'r+b')  # a21.txt必须存在，否则出错，读写方式打开
      # f1.write('123') #错误：只能写字节数据
      print(f1.read(3))  # 从文件中读取3个字符
      f1.write(b'789')
    f1.seek(0, os.SEEK_END)  # 定位到文件结束位置
      b = bytearray(10)  # 从文件中读取10个字节
      print(f1.readinto(b))  # 从文件中读取,因为指针已经在文件末尾，返回0
    f1.seek(0, os.SEEK_SET)  # 定位到文件开始位置
      print(f1.read())  # 读取文件的全部内容
      f1.seek(-15, os.SEEK_CUR)  # 文件指针从当前位置（文件末尾）向前移动15个字节
      print(f1.readinto(b))  # 读出10个字节
      print(b)
except IOError as e:
    print(e)
    exit()
finally:
    f1.close()
```

其他文件读写：CSV文件的读出

打开文件时mode参数设置

和读其它文件时一样

csv.reader对象

功能：从csv文件或者list对象中读取数据。 其构造函数为：

csv.reader(csvfile,dialect=’ excel’ ,**fmtparams)

csv.reader常用属性

若 ***fr*** 为csv.reader对象，则 ***fr.line_num*** 返回读入的行数

```
# demo0707.py
BasePath = 'c:\\data\\'
import csv

with open(BasePath + 'table.csv', 'r',
encoding="utf-8") as csvfile:
    f_csv = csv.reader(csvfile, delimiter=',')
    print('--------Header--------')
    print(next(f_csv))  # f_csv是一个迭代对象
      print('--------Data----------')
    for row in f_csv:
        print(row)
```

table.csv内容

```
姓名,语文 ,数学
张三,89,95
李四,92,92
王五,79,98
```

其他文件读写：CSV文件的写入

打开文件时mode参数设置

和写其它文件时的参数一样

csv.writer对象

功能：把列表对象数据写入到CSV文件中，其构造函数语法为： ***csv.writer(csvfile,dialect='excel',**fmtparams)***

csv.writer常用属性

csvwriter.writerow(row)：写入一行数据的方法

csvwriter.writerows(rows)：写入多行数据的方法

csvwriter.dialect：返回其dialect的只读属性

```
# demo0708.py
BasePath = 'c:\\data\\'
import csv


def writecsv1(filepath):
    with open(filepath, 'a', newline='', encoding="utf-8")
as csvfile:
        rows = [('赵六', 86, 97), ('孙七', 95, 95)]
        f_csv = csv.writer(csvfile)
        f_csv.writerows(rows)  # 在文件末尾写入两行数
据。

if __name__ == '__main__':
        writecsv1(BasePath + 'table.csv')
```

其他文件读写：OS模块访问文件

文件描述符

操作系统中用一个整数代表文件： 0：标准输入 1：标准输出 2：标准错误

打开文件 *os.open(file,flags[,mode])*

Flags选项

os.O_RDONLY	以只读方式打开
os.O_WRONLY	以只写方式打开
os.O_RDWR	以读写方式打开
os.O_APPEND	以添加方式打开，新内容写到文件末尾
os.O_CREAT	以新建方式打开
os.O_EXCL	如果文件已存在，则返回错误
os.O_TRUNC	将已存在文件的内容清空再打开

文件指针定位

os.lseek(fd,pos,how)

pos：一个整数，表示偏移位置

how：*os.SEEK_SET* 表示文件开始，*os.SEEK_CUR* 表示当前指针位置，*os.SEEK_END*表示文件末尾

写入/读取数据

os.read(fd,n)：从fd中读取之多n个字符，返回bytestring 对象。

os.write(fd,str)：字符串str写入文件标识符fd连接的文 件，返回实际写入字节数。

os.fsync(fd)：将缓冲区中数据立即写入文件。

关闭文件

os.close(fd)

其他文件读写：OS模块访问文件

```
# demo0710.py
BasePath = 'c:\\data\\'
import os

fd = os.open(BasePath + 'os.dat'
             , os.O_RDWR | os.O_CREAT)
os.write(fd, b'123')  # 写入'123'
os.lseek(fd, -3, os.SEEK_CUR)  # 从当前位置往回3个字符的位置
print(os.read(fd, 2))  # 输出的内容为b'12'
os.write(fd, b'abcdef')  # 从第三个字符的位置开始写入新的内容
os.lseek(fd, 0, os.SEEK_SET)  # 定位到文件开始位置
print(os.read(fd, 100))  # 输出前100个字符的内容
```

对象序列化

对象的序列化（*pickling*）

内存对象→文件存储或者网络传输对象

对象的反序列化（*unpickling*）

文件中存储或者网络传输对象→内存对象

作用

对象的持久保存和跨平台传输

```
# demo0811.py 序列化
import pickle

BasePath = 'c:\\data\\'
with open(BasePath + 'object.dat'
          ,
          'wb') as filedump:
    obj1 = 'Test'
    obj2 = ('Justfortest'
            , dict(name='张三',
                   score=dict(语文=100,
数学=123, 英语=88)))
    pickle.dump(obj1, filedump)
    pickle.dump(obj2, filedump)
```

```
#反序列化
BasePath = 'c:\\data\\'
import pickle


with open(BasePath + 'object.dat'
          , 'rb') as fileload:
    obj1 = pickle.load(fileload)
    obj2 = pickle.load(fileload)
    print(type(obj1), str(obj1))
    print(type(obj2), str(obj2))
```

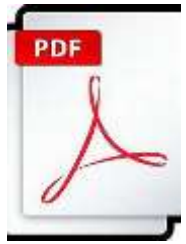
常用办公文件格式的处理

Python除了内置库函数提供的文件访问能力，还可以通过丰富的第三方库操作更多类型的文件。

常见的文件类型，比如办公软件office系列文件*word*、*excel*和*ppt*，文档发布最常用的 *pdf* 等等。

典型的应用场景有

- ✓ 从数据库内容中生成定制化的格式化文档
- ✓ 工作过程根据流程数据自动生成文档
- ✓ 特别是非windows平台上进行此类文档的处理



一般来说，文档处理需求有这样的特点

- ✓ 免于简单重复的手工操作
- ✓ 规范化的批量处理
- ✓ 跨平台、服务端的文档编辑

常用办公文件格式的处理：EXCEL

1）一个 Excel 电子表格文档称为一个工作簿。每个工作簿可以包含多个表（也称为工作表 *sheet*）。
2）用户当前查看的表（或关闭 Excel 前最后查看的表），称为活动表。
3）每个表都有一些列和一些行（地址是从 1 开始的 数字）。在特定行和列的方格称为单元格。在单元格 里面，可以存入数据。
4）默认列名，是从 *A* 开始的26个字母，用完之后*A?* 继续26个，*B?*继续26个...两位用完用三位。
5) *Excel2003(.xls)*文件：最大256(2的8次方)列，最 大65536(2的16次方)行。*Excel2007(.xlsx)* 文件：最 大16384(2的14次方)列，最大1048576(2的20次方)行

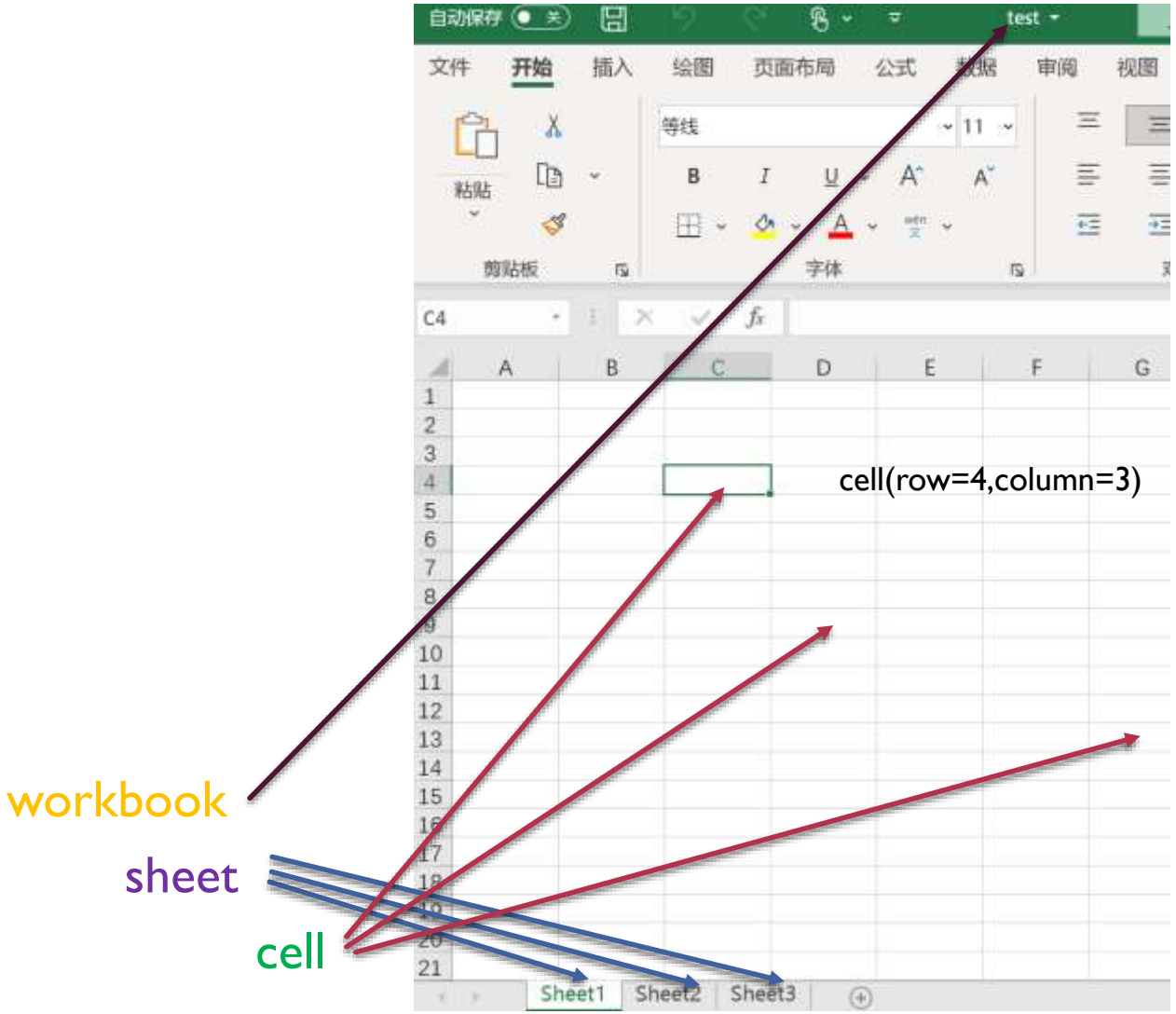
1）Python对Excel的读写主要有*xlrd*、*xlwt*、*xlutils*、*openpyxl*、*xlsxwriter*、*pandas*等几种方法。

- ✓ ***xlrd***:主要是用来读取excel文件。
- ✓ ***xlwt:***主要是用来写excel文件。
- ✓ ***xlutils***:结合*xlrd*可以达到修改excel文件目的。
- ✓ ***xlsxwriter***:可以写excel文件并加上图表。
- ✓ ***openpyxl:***功能强大的excel文件处理库。
- ✓ ***pandas***:功能强大的excel数据处理库。（侧重于表格 数据处理和分析，后续讲）

2） 安装***openpyxl*** 模块。

常用办公文件格式的处理：EXCEL

- ✓ ***Openpyxl*** 几乎可以实现所有的 *Excel* 功能，而且接口清晰，简单易用，文档丰富，学习成本相对较低
- ✓ ***Openpyxl*** 用*workbook-sheet-cell* 的模式对.xlsx文件进行读写改操作，并可调整表格样式



常用办公文件格式的处理：EXCEL-工作表读取

1）***openpyxl.load_workbook()***。函数接受文件名，返回一个 *workbook* 数据类型的值。打开文件生成*workbook*对象，相当于特殊的文件对象。

2）***sheetnames***。可以取得工作簿中所有工作 表名的列表。

3）***wb['XX']***。通过*sheet*名字，来获取工作表。

4）***wb.active***。取得 工作簿的活动表。获得活动表后，可以通过 *title* 属性取得它的名称。

序号	学号	姓名	班号	辅导员	年级	校区	未返校原因（疫情原因/因病/其他原因需注明）
1		张艺隆	2020211303	陈宇寒	2020	西土城校区	疫情原因
2		贺政苟	2020211303	陈宇寒	2020	西土城校区	疫情原因
3		徐子然	2020211303	陈宇寒	2020	西土城校区	疫情原因
4		刘宇欣	2020211304	陈宇寒	2020	西土城校区	疫情原因
5		王辉仕	2020211305	陈宇寒	2020	西土城校区	疫情原因
6		潘婷	2020211305	陈宇寒	2020	西土城校区	疫情原因

```
import openpyxl

wb=openpyxl.load_workbook(r'C:\工作目录\北邮\北邮教学\2022 网工\本科生延期返校名单.xlsx')
wb_sheets=wb.sheetnames
print(wb_sheets)
sheet=wb[wb_sheets[0]]
print(sheet.title)
sheet=wb.active
print(sheet.title)
```

常用办公文件格式的处理：EXCEL-单元格读取

1）*sheet['A1']*。读取该*sheet*页的第“A”列，第“1”行，返回为一个Cell对象。

2）*Cell*对象有*value*、*row*、*column*和*coordinate*属性，分别代表值、行、列、位置信息。

3）*sheet.cell(row=i, column=j)*，也可以指定对应行、列的单元格。

4）*column_index_from_string()*。从字母转换到数字。 *get_column_letter()*。数字转换到字母。

序号	学号	姓名	班号	辅导员	年级	校区	未返校原因（疫情原因/因病/其他原因需注明）
1		张艺隆	2020211303	陈宇寒	2020	西土城校区	疫情原因
2		贺政苟	2020211303	陈宇寒	2020	西土城校区	疫情原因
3		徐子然	2020211303	陈宇寒	2020	西土城校区	疫情原因
4		刘宇欣	2020211304	陈宇寒	2020	西土城校区	疫情原因
5		王辉仕	2020211305	陈宇寒	2020	西土城校区	疫情原因
6		潘婷	2020211305	陈宇寒	2020	西土城校区	疫情原因

```
cell_1=sheet['C2']
print(cell_1.value,cell_1.row,cell_1.column,cell_1.coordinate,sep='##')
cell_2=sheet.cell(6,3)
print(cell_2.value,cell_2.row,cell_2.column,cell_2.coordinate,sep='##')
for i in range(2,9,2):
    print(sheet.cell(row=i,column=3).value)

from openpyxl.utils import column_index_from_string,get_column_letter
print('column index for AA is {}\n'.format(column_index_from_string('AA')))
print('letter for column index 30 is {}'.format(get_column_letter(30)))
```

常用办公文件格式的处理：EXCEL-工作表切片

1）可以将*Worksheet*对象切片，取得电子 表格中一行、一列或一个矩形区域中的所 有Cell对象。

2）*sheet.rows*。按行进行数据读取，1行 1行处理：*A1、B1、C1...*

3）*sheet.columns*。按列进行数据读取， 1列1列处理：*A1、A2、A3...*

序号	学号	姓名	班号	辅导员	年级	校区	未返校原因（疫情原因/因病/其他原因需注明）
1		张艺隆	2020211303	陈宇寒	2020	西土城校区	疫情原因
2		贺政苟	2020211303	陈宇寒	2020	西土城校区	疫情原因
3		徐子然	2020211303	陈宇寒	2020	西土城校区	疫情原因
4		刘宇欣	2020211304	陈宇寒	2020	西土城校区	疫情原因
5		王辉仕	2020211305	陈宇寒	2020	西土城校区	疫情原因
6		潘婷	2020211305	陈宇寒	2020	西土城校区	疫情原因

```
highlight_cells=sheet['C2':'E7']

for row_cells in highlight_cells:
    for cell in row_cells:
        print(cell.coordinate,cell.value)

for row in sheet.rows:
    for cell in row:
        print(cell.coordinate,cell.value)
for col in sheet.columns:
    for cell in col:
        print(cell.coordinate,cell.value)
```

常用办公文件格式的处理：EXCEL-读取示例

1）读取班级号的列。

2）*统计有延期返校同学的班级、每个班级有多少同学延期返校*

3）*统计总数并与总行数做比对*

```
stat_column=[each[3].value for each in sheet.rows]
stat_class={}

print('sheet contains {} rows', len(stat_column))

for i in range(1,len(stat_column)):
    stat_class[stat_column[i]]=
                    stat_class.get(stat_column[i],0) +1

print('class with students at home:',stat_class)

from functools import reduce
print(reduce(lambda x,y:x+y, stat_class.values()))
```

```
sheet contains {} rows 242
class with students at home: {2020211303: 3, 2020211304: 1, 2020211305: 4, 2020211306: 3, 2020211307: 2,
2020211308: 1, 2019211308: 6, 2019211309: 1, 2019211310: 4, 2019211311: 3, 2019211312: 3, 2020211309: 3,
2020211310: 1, 2020211311: 3, 2020211312: 1, 2020211313: 2, 2020211323: 1, 2022281301: 4, 2021211311: 3,
2021211312: 2, 2021211316: 1, 2021211317: 2, 2021211320: 3, 2021211319: 1, 2022211315: 4, 2022211318: 5,
2022211316: 3, 2022211319: 3, 2022211317: 3, 2022211313: 5, 2022211314: 2, 2020211301: 1, 2020211302: 2,
2020211314: 3, 2020211315: 5, 2020211321: 4, 2020211322: 3, '#N/A': 5, 2019211302: 5, 2019211303: 3,
2019211304: 5, 2019211305: 3, 2019211306: 4, 2019211301: 1, 2020211318: 2, 2020211319: 1, 2020211320: 1,
2021211301: 2, 2021211302: 1, 2021211303: 3, 2021211321: 3, 2021211322: 1, 2021281301: 3, 2019211501: 4,
2019211502: 3, 2019211503: 2, 2019211504: 2, 2019211505: 3, 2022211301: 4, 2022211302: 3, 2022211303: 3,
2022211304: 4, 2022211305: 5, 2019211313: 5, 2019211314: 4, 2019211315: 6, 2019211316: 3, 2019211317: 3,
2019211318: 5, 2021211304: 1, 2021211307: 4, 2021211308: 1, 2021211309: 1, 2022211306: 5, 2022211308: 4,
2022211311: 4, 2022211309: 2, 2022211307: 5, 2022211310: 6, 2022211312: 5}
241
```

常用办公文件格式的处理：EXCEL-创建、维护工作表

1）*openpyxl.Workbook()*。创建新的空 *Workbook* 对象。

2）*wb['Sheet1']*。按名字指定*sheet* 页。

3）*sheet.title*。修改*sheet* 名字。

4）*create_sheet()*、*remove()*。创建和删除*sheet* 页。

5）移动*sheet* 页。

wb.move_sheet(wb[‘工资表’],1) #把指定的*sheet* 页，向后移动1个位置。

6）复制*sheet* 页。

wb.copy_worksheet(wb['工资表’]) #把'工资表'sheet复制1份，默认命名为'工资表Copy'

```
import openpyxl

wb=openpyxl.Workbook()

wb_sheets=wb.sheetnames
print(wb_sheets)

sheet=wb[wb_sheets[0]]
sheet.title='花名册'
wb.create_sheet('工资表')
wb.create_sheet('temp',1)
print(wb.sheetnames)

wb.remove(wb['temp'])
print(wb.sheetnames)

wb.copy_worksheet(wb['工资表'])
wb.move_sheet(wb['工资表'],1)
print(wb.sheetnames)

wb.save(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据\sample_create_excel.xlsx')
wb.close()
```

常用办公文件格式的处理：EXCEL-写入数据

1）写入单元格。
sheet['A1'] = value
sheet.cell(3,1).value = value
2）可以通过循环遍历一个区域或全部行和列，进行批量的数据复制。
注意：行、列，都是从1开始；循环变量一般是从0开始。如右边的薪水列号，和复制语句中的*row+1*、*col+1*。 另，此处跳过第5列，由于后面正常复制了第6列且列号是6，所以导致第5列在目标文 件夹中也存在，但为空列。如果彻底不希望被跳过的列出现，应当每跳过1列，记一个 数*x+1*，修改“*col+1*”为“*col+1-x*”。

```
import openpyxl

wb = openpyxl.load_workbook(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据
\sample_create_excel.xlsx')

sheet = wb['花名册']
sheet['A2'] = '杨过'
sheet.cell(3, 1).value = '郭襄'

wb_source = openpyxl.load_workbook(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据
\sample_excel_data.xlsx')
sheet_source = wb_source['Sheet1']
for row in range(sheet_source.max_row):
    for col in range(sheet_source.max_column):
        if col == 4:  # skip column 4
            continue
        sheet.cell(row + 1, col + 1).value = sheet_source.cell(row + 1, col + 1).value

wb.save(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据\sample_create_excel.xlsx')
wb.close()
wb_source.close()
```

常用办公文件格式的处理：EXCEL-批注和公式

1）*openpyxl.comments* 中的 ***Comment*** 方法，给单元格增加批注。

——可以用于检查数据合法性后， 把检测违规的单元格数据非法情况，描述在批注里。供用户方便确认和修改。

注意：低于35000的单元格， 没有写入该标注。

comment=None：去掉批注。

2）通过*openpyxl* 模块，可以直接给单元格**写入公式。**

备注：通过该方法，也间接相当于通过Python调用了Excel里的函数，不用增加学习成本就可以使用大量在Excel 中已经习惯的功能。

```
# set comments
wb = openpyxl.load_workbook(r'C:\工作目录\北邮\北邮教学\2022网工\示例数据\sample_excel_data.xlsx')
sheet = wb['Sheet1']
sheet['A2'].comment = Comment('已打过疫苗', 'BUPT')

for row in range(sheet.max_row):
    if row == 0: continue
    if float(sheet.cell(row + 1, 5).value) >= 35000:
        sheet.cell(row + 1, 5).comment = Comment('高收入人群', 'BUPT')

# use excel functions
row_num = sheet.max_row
sheet.cell(row_num + 1, 5).value = f'=SUM(E2:E{row_num})'
sheet.cell(row_num + 2, 5).value = f'=MIN(E2:E{row_num})'
sheet.cell(row_num + 3, 5).value = f'=MAX(E2:E{row_num})'
```

常用办公文件格式的处理：EXCEL-设置样式参数

1） *openpyxl* 处理Excel文件中单元格样式，总共有六个属性类。

a) *font* (字体类，可设置字号、字体颜色、下划线等)；

b) *fill* (填充类，可设置单元格填充颜色等)；

c) *border* （边框类，可以设置单元格各种类型的边框）；

d) *alignment* (位置类、可以设置单元格内数据各种对齐方式)；

e) *number_format* (格式类，可以设置单元格内各种类型的数据格式)；

f) *protection* (保护类，可以设置单元格写保护等)。

```
# patterns
from openpyxl.styles import PatternFill, Border, Side, Alignment, Protection, Font

row_head = sheet.row_dimensions[1]
row_head.height = 20

for col in range(sheet.max_column):
    sheet.cell(1, col + 1).fill = PatternFill(fill_type='solid', fgColor='92D050')
    sheet.cell(1, col + 1).font = Font(name='隶书', size=13, bold=True)
    sheet.cell(1, col + 1).alignment = Alignment(horizontal='center', vertical='center')

for col2 in range(sheet.max_column):
    sheet.cell(1, col2 + 1).border = Border(top=Side(border_style='thick', color='000000'))
    sheet.cell(sheet.max_row, col2 + 1).border = Border(bottom=Side(border_style='thick',
color='000000'))

for row in range(sheet.max_row):
    if row == 0:
        sheet.cell(1, 1).border = Border(left=Side(border_style='thick', color='000000'),
                                         top=Side(border_style='thick', color='000000'))
        sheet.cell(1, sheet.max_column).border = Border(right=Side(border_style='thick', color='000000'),
                                                        top=Side(border_style='thick', color='000000'))
    elif row == sheet.max_row - 1:
        sheet.cell(row + 1, 1).border = Border(left=Side(border_style='thick', color='000000'),
                                               bottom=Side(border_style='thick', color='000000'))
        sheet.cell(row + 1, sheet.max_column).border = Border(right=Side(border_style='thick',
color='000000'),
                                                              bottom=Side(border_style='thick', color='000000'))
    else:
        sheet.cell(row + 1, 1).border = Border(left=Side(border_style='thick', color='000000'))
        sheet.cell(row + 1, sheet.max_column).border = Border(right=Side(border_style='thick',
color='000000'))
```

常用办公文件格式的处理：EXCEL-创建图表

1）通过*openpyxl*模块，也可以在Python里面给*Excel*文件插入图表。

备注：直接使用表格里面的列作为数据，或者其他图表样式，需要用到的时候自己再研究。


```
# Charts
from openpyxl.chart import PieChart, Reference

salary_chart = PieChart()
salary_level = {'高收入': (35000, 100000), '中等收入': (20000, 35000), '普通收入': (0, 20000)}
salary_stats = {'高收入': 0, '中等收入': 0, '普通收入': 0}

for row in range(sheet.max_row):
    if row == 0: continue
    for k, v in salary_level.items():
        if v[0] <= float(sheet.cell(row + 1, 5).value) < v[1]:
            salary_stats[k] += 1

salary_row = 2
salary_col = sheet.max_column + 1
for k, v in salary_stats.items():
    sheet.cell(salary_row, salary_col).value = k
    sheet.cell(salary_row, salary_col + 1).value = v
    salary_row += 1

labels = Reference(sheet, min_col=salary_col, min_row=2, max_row=4)
data = Reference(sheet, min_col=salary_col + 1, min_row=1, max_row=4)
salary_chart.add_data(data, titles_from_data=True)
salary_chart.set_categories(labels)
salary_chart.title = '毕业生薪资分析'

sheet.add_chart(salary_chart, 'H1')
```

常用办公文件格式的处理：EXCEL-其他

1）***merge_cells***。

sheet.merge_cells('C5:D5')：合并*C5和D5*两个单元格。

sheet.merge_cells('A1:D3')：合并*A1：D3*的区域。

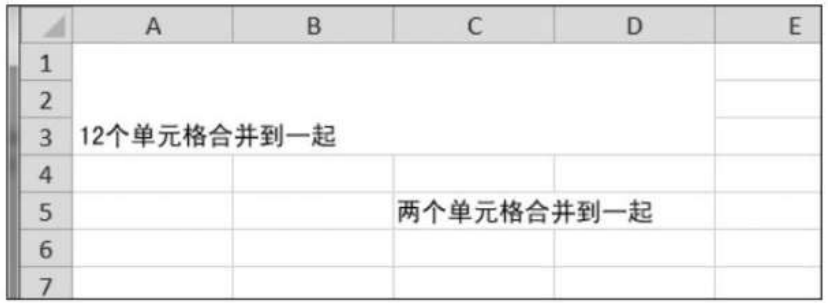
2）***unmerge_cells***。

sheet.unmerge_cells('A1:D3'）。

3）每个*Worksheet*对象都有一个***freeze_panes***属性， 可以设置为一个*Cell*对象或一个单元格坐标的字符串。 请注意，单元格上边的所有行和左边的所有列都会冻结，但单元格所在的行和列不会冻结。

sheet.freeze_panes = 'A2 ' 冻结第1行。

sheet.freeze_panes ='B1 ' 冻结A列。



常用办公文件格式的处理：EXCEL-模拟高考赋分

自2020年起，北京实行“3+3”新高考制度

常用办公文件格式的处理：EXCEL-模拟高考赋分

北京高考赋分制等级表

等	A					B					C					D					E
比例	15%					40%					30%					14%					1%
级	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5	C1	C2	C3	C4	C5	D1	D2	D3	D4	D5	E
比例	1%	2%	3%	4%	5%	7%	8%	9%	8%	8%	7%	6%	6%	6%	5%	4%	4%	3%	2%	1%	1%
分数	100	97	94	91	88	85	82	79	76	73	70	67	64	61	58	55	52	49	46	43	40

常用办公文件格式的处理：EXCEL-模拟高考赋分 问题要求

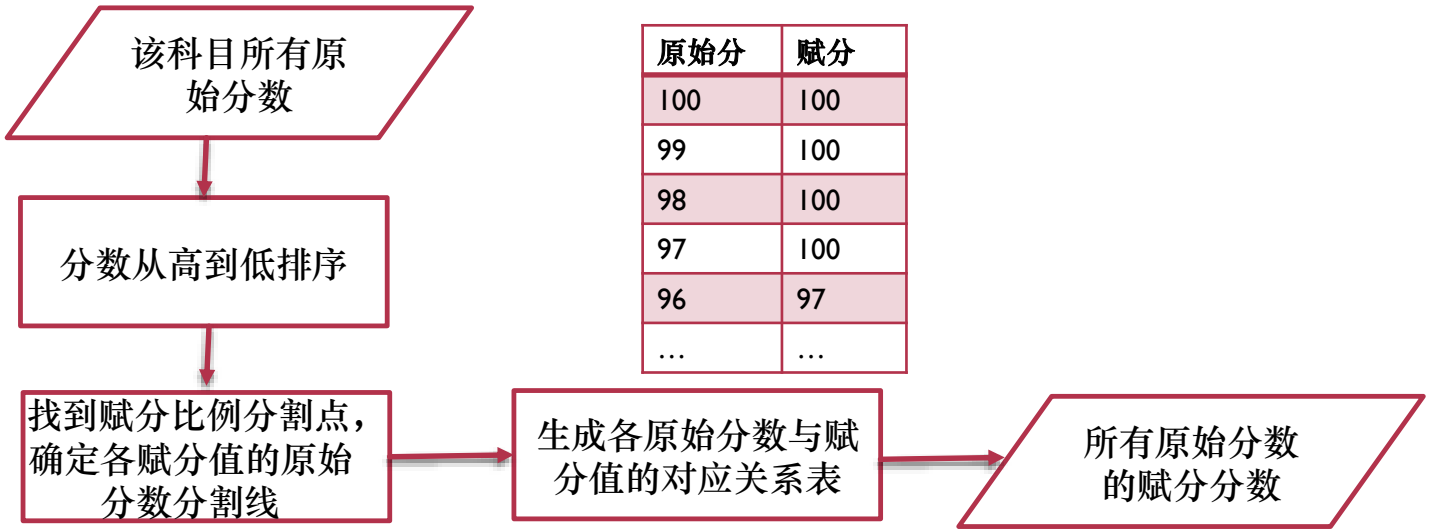
- ✓ 输入：Excel表格，存储准考证号及各科原始成绩
- ✓ 输出：赋分科目的赋分成绩，写入Excel表格

	A	B	C	D	E	F	G	H	I	J
1	准考证号	数学	数学	英语	物理	化学	生物	政治	历史	地理
2	101010101	69	49	42	28			24		36
3	101010102	95	11	0	73			2	20	
4	101010103	8	67	71		96		78	63	
5	101010104	70	72	68	48	64			41	
6	101010105	17	41	97	54		72	59		
7	101010106	99	71	47			55	69		83
8	101010107	39	0	18	10			11		97
9	101010108	53	91	32	6	66				41
10	101010109	22	70	74		64			73	38
11	101010110	74	1	81		17	14			55
12	101010111	91	36	40	44	44			20	
13	101010112	35	49	11			61	71		58
14	101010113	39	60	20		88		98		50
15	101010114	10	18	80		70		42	35	
16	101010115	8	93	54			3		91	74
17	101010116	90	16	76		80		83		41
18	101010117	41	63	6	18	75				85
19	101010118	93	97	93	44	76	3			

常用办公文件格式的处理：EXCEL-模拟高考赋分 核心实现流程

本题目中，核心流程是对每一个赋分科目，实现从原始分到赋分的转换

比例	赋分值	原始分数分割线
1%	100	97
2%（1~3%）	97	95
3%（6~9%）	94	91
…	…	….



原始分	赋分
100	100
99	100
98	100
97	100
96	97
…	…

常用办公文件格式的处理：EXCEL-模拟高考赋分

等级	占比	排名	分数
A1	1%	A1 ≥ 1%	100
A2	2%	1% < A2 ≤ 3%	97
A3	3%	3% < A3 ≤ 6%	94
A4	4%	6% < A4 ≤ 10%	91
A5	5%	10% < A5 ≤ 15%	88
B1	7%	15% < B1 ≤ 22%	85
B2	8%	22% < B2 ≤ 30%	82
B3	9%	30% < B3 ≤ 39%	79
B4	8%	39% < B4 ≤ 47%	76
B5	8%	47% < B5 ≤ 55%	73
C1	7%	55% < C1 ≤ 62%	70
C2	6%	62% < C2 ≤ 68%	67
C3	6%	68% < C3 ≤ 74%	64
C4	6%	74% < C4 ≤ 80%	61
C5	5%	81% < C5 ≤ 85%	58
D1	4%	86% < D1 ≤ 89%	55
D2	4%	89% < D2 ≤ 93%	52
D3	3%	93% < D3 ≤ 96%	49
D4	2%	96% < D4 ≤ 98%	46
D5	1%	98% < D5 ≤ 99%	43
E	1%	99% < E ≤ 100%	40

```
grading_dict = {100: 1, 97: 3, 94: 6, 91: 10, 88: 15, 85: 22, 82: 30, 79: 39, 76: 47,
                73: 55, 70: 62, 67: 68, 64: 74, 61: 80, 58: 85, 55: 89, 52: 93,
                49:96, 46: 98, 43: 99, 40: 100}


def trans_grading(scores):
    """ translate grading table to course specific grading slicing """
    dict2 = {}
    scs = sorted(scores, reverse=True)
    l_g = len(scs)
    for k in grading_dict.keys():
        pos = round(l_g * grading_dict[k] / 100)
        if pos < l_g:
            dict2[k] = scs[pos]
        else:
            dict2[k] = 0
    return dict2
```

常用办公文件格式的处理：EXCEL-模拟高考赋分 生成赋分字典

赋分	原始分数分割线
100	97
97	95
94	91
…	….

原始分	赋分
100	100
99	100
98	100
97	100
96	97
…	…

```
def trans_score(grades):
    """ translate original score to grading score """
    dict4 = {}
    for i in range(100, -1, -1):
        dict4[i] = 0
        for k in grades.keys():
            if i >= grades[k] and dict4[i] < k:
                dict4[i] = k
    return dict4
```

常用办公文件格式的处理：EXCEL-模拟高考赋分 准考证号和考生数

北京市公布高考考点考场安排，2020年北京设17个考区，132个考点学校，2867个考场。其中，备用考点共22个。

准考证是参加高考的入场凭证。**准考证号**由9位数字**组成**，左边5位为考点代号，第6、7位为考场号，第8、9位为考场内座次号。

模拟准考证号格式如下：

AAABBCCDD

- AA：考区号，101~117
- BB：考区内考点号，设各考区考点数：5~20以内随机数
- CC：考场号，设各考点考场数：10~30以内随机数
- DD：考场内座次号：01~20

常用办公文件格式的处理：EXCEL-模拟高考赋分 生成准考证号

AAA：101~117，固定

BB：针对每个AAA，生成一个5~20之内的随机数，存入字典{ ‘AAA’ :BB}

AAA01~AAABB

CC：针对每个AAABB，生成一个10~30内的随机数，存入字典{ ‘AAABB’ : CC}

DD：对于每个考点AAABB，CC最大的考场座位数为01~20之间的随机数，其它考场座位数为20个。生成字典{ ‘AAABBCC’ ： XX}，其中：XX=20或01~20之间的随机数

	A	B	C	D
1	准考证号			
2	101010101			
3	101010102			
4	101010103			
5	101010104			
6	101010105			
7	101010106			
8	101010107			
9	101010108			
10	101010109			
11	101010110			
12	101010111			
13	101010112			
14	101010113			
15	101010114			
16	101010115			
17	101010116			
18	101010117			

常用办公文件格式的处理：EXCEL-模拟高考赋分 生成课程名

	A	B	C	D	E	F	G	H	I	J
1	准考证号	语文	数学	英语	物理	化学	生物	政治	历史	地理
2	101010101									
3	101010102									
4	101010103									
5	101010104									
6	101010105									
7	101010106									
8	101010107									
9	101010108									
10	101010109									
11	101010110									
12	101010111									

常用办公文件格式的处理：EXCEL-模拟高考赋分 生成原始分数

```
def generate_ori_scores(file_name, sheet_name):
    """ generate original scores for all 6 courses """
    j = 0
    wb = load_workbook(file_name)
    ws = wb[sheet_name]
    num_of_students = ws.max_row - 1
    score_list = np.random.randint(low=0, high=150, size=num_of_students * 3)  # first 3 courses
    for r in range(2, num_of_students + 2):
        for c in range(2, 11):
            ws.cell(r, c).value = ''
    for r in range(2, num_of_students + 2):
        for c in range(2, 5):
            ws.cell(row=r, column=c, value=score_list[j])
            j = j + 1
    score_list = np.random.randint(low=0, high=100, size=num_of_students * 3)  # 3 optional courses
    j = 0
    for r in range(2, num_of_students + 2):
        choose = np.random.choice(6, 3, replace=False)
        for c in choose:
            ws.cell(row=r, column=c + 5, value=score_list[j])
            j = j + 1
    wb.save(file_name)
    wb.close()
```

常用办公文件格式的处理：EXCEL-模拟高考赋分

	A	B	C	D	E	F	G	H	I	J
1	准考证号	语文	数学	英语	物理	化学	生物	政治	历史	地理
2	101010101	97	59	27			73		89	85
3	101010102	104	78	117			86		90	27
4	101010103	16	105	72		11	18		6	
5	101010104	56	127	8			90		81	28
6	101010105	54	101	109		30	91			36
7	101010106	56	32	17			74	79	16	
8	101010107	121	78	86	89	13	53			
9	101010108	26	31	108	7	82			60	
10	101010109	23	92	99		20	50			70
11	101010110	102	108	83		30	2		87	

常用办公文件格式的处理： EXCEL-模拟高考赋分 生成赋分

```
def score_by_column(file_name, sheet_name, score_cols):
    """ score_cols: {ori_col_index:grad_col_index} """
    wb = openpyxl.load_workbook(file_name)
    ws = wb[sheet_name]
    rows = ws.max_row
    for ori, grad in score_cols.items():
        col_data = []
        for i in range(2, rows + 1):
            cell_value = ws.cell(row=i, column=ori).value
            if cell_value != None and cell_value != -1:
                col_data.append(cell_value)  # valid score list, neither blank(not chosen) nor -1(missing exam)
        degree_dict = trans_grading(col_data)  # generate grading rule table
        trans_dict = trans_score(degree_dict)  # generate grading score

        sub_name = ws.cell(row=1, column=ori).value
        ws.cell(row=1, column=grad, value=sub_name + '赋分')  # set column name
        for r in range(2, rows + 1):
            ori_score = ws.cell(row=r, column=ori).value
            if isinstance(ori_score, int):
                ws.cell(row=r, column=grad, value=trans_dict[ori_score])
    wb.save(file_name)
    wb.close()
```

常用办公文件格式的处理：EXCEL-模拟高考赋分

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	准考证号	语文	数学	英语	物理	化学	生物	政治	历史	地理	物理赋分	化学赋分	生物赋分	政治赋分	历史赋分	地理赋分
2	10101010	97	59	27			73		89	85			82		88	88
3	10101010	104	78	117			86		90	27			88		91	64
4	10101010	16	105	72		11	18		6			55	58		52	
5	10101010	56	127	8			90		81	28			91		85	64
6	10101010	54	101	109		30	91			36		64	91			67
7	10101010	56	32	17			74	79	16				82	85	58	
8	10101010	121	78	86	89	13	53				88	55	76			
9	10101010	26	31	108	7	82			60		52	85			76	
10	10101010	23	92	99		20	50			70		61	73			82
11	10101011	102	108	83		30	2		87			64	46		88	
12	10101011	0	100	139			1	93	32				46	94	67	
13	10101011	106	15	53	28		26		28		64		64		64	
14	10101011	38	95	111	86		1		33		88		46		67	
15	10101011	17	35	24	42	83				9	70	85				52
16	10101011	38	127	73		45	45	13				73	73	55		
17	10101011	16	105	34			11	96		1			55	94		46
18	10101011	145	18	32				24	79	45				61	85	73
19	10101011	145	116	149		71	68	38				82	79	70		
20	10101011	38	61	78		97	75			47		97	82			73
21	10101012	36	30	84		83	75		9			85	82		52	
22	10101020	43	54	121	35	26			81		67	64			85	
23	10101020	68	69	69		56	50			1		76	73			46
24	10101020	149	74	74		64	52		48			79	76		73	
25	10101020	34	118	128		57	39			52		76	70			76

常用办公文件格式的处理：WORD-文档对象模型

1）利用***python-docx*** 模块处理*word* 文档。模块会把*word* 文档中的段落、文本、字体等都看做对象。

2）安装时：*pip install python-docx*。导入时：*import docx*。

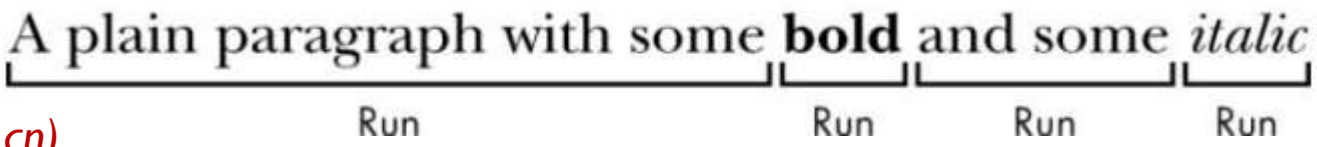
3）和纯文本相比，*.docx* 文件有很多结构。这些结构在*python-docx* 中用3种不同的类型来表示。

a）在最高一层，***Document*** 对象表示整个文档。

b）*Document* 对象包含一个 ***Paragraph*** 对象的列表，表示文档中的段落（用户在 *Word* 文档中输入时，如 果按下回车，新的段落就开始了）。

c）每个 *Paragraph* 对象都包含一个 ***Run*** 对象的列表。

4）*Word* 文档中的文本不仅仅是字符串。它包含与之相关的字体、大小、颜色和其他样式信息。在*Word* 中，样式是这些属性的集合。一个 *Run* 对象是相同样式文本的延续。当文本样式**发生改变**时，就需要一个新的 *Run* 对象。下面的一句话的段落中，就有*4* 个 *Run* 对象。



官方文档：python 文档 — python-docx 0.8.11 文档 (osgeo.cn)

常用办公文件格式的处理：WORD-读取

1）***docx.Document(path)***。打开*Word*文件， 返回***Document*** 对象。
2）***paragraphs***。*Document* 对象的段落**列表**， 是一个由段落对象作为元素的列表。
3）***runs***。每个*Paragraph* 对象也有一个*runs* 属性，它是***Run*** 对象的**列表**。

```
This document has 30 paragraphs.

传统工艺三维纹样曲面的二维化是计算机视觉中亟待解决的问题，但目前的多数研究只关注三维空间或者二维平面的问题，
缺少三维纹样曲面二维化的相关研究。因此，传统工艺三维纹样曲面的二维化面临着较大的挑战。本文以传统工艺三维纹
样曲面为研究对象，开展二维化方法的研究。本文率先提出了基于传统工艺三维纹样曲面的二维化算法，并搭建基于传统
工艺曲面纹样的展平系统，研究内容和创新点主要包括：
Paragraph 6 has 13 segments.
（
2
['基于传统工艺曲面纹样展平系统的研发与实现', '', '摘　要', '', '传统工艺三维纹样曲面的二维化是计算机视
觉中亟待解决的问题，但目前的多数研究只关注三维空间或者二维平面的问题，缺少三维纹样曲面二维化的相关研究。因
此，传统工艺三维纹样曲面的二维化面临着较大的挑战。本文以传统工艺三维纹样曲面为研究对象，开展二维化方法的研
究。本文率先提出了基于传统工艺三维纹样曲面的二维化算法，并搭建基于传统工艺曲面纹样的展平系统，研究内容和创
新点主要包括：', '（1）提出三维纹样曲面二维化拼接方法。三维纹样曲面经过曲面展开及图像拼接得到完整的二维化
```

```python
import docx

doc=docx.Document(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据
\word_sample_read.docx')
print(f'This document has {len(doc.paragraphs)} paragraphs.')
print(doc.paragraphs[1].text)
print(doc.paragraphs[4].text)

print(f'Paragraph 6 has {len(doc.paragraphs[6].runs)} segments.')
print(doc.paragraphs[6].runs[0].text)
print(doc.paragraphs[6].runs[1].text)

full_text=[]
for para in doc.paragraphs:
    full_text.append(para.text)
print(full_text)
```

常用办公文件格式的处理：WORD-写入

1）***docx.Document()***。返回一个新的、空白的*WordDocument* 对象。
2）*WordDocument* 的***add_paragraph()*** 方法 将一段新文本添加到文档中，并返回添加的*Paragraph* 对象的引用。*Paragraph* 对象 的***add_run()*** 方法，向它传入一个字符串。
3）在添加完文本之后，*Document* 对象的 ***save(path)*** 方法传入一个文件名字符串， 将 *Document* 对象保存到文件。

自动添加的第一段文字。

自动添加的第二段文字。继续添加第二段文字。

```
import docx

doc=docx.Document()

para_1=doc.add_paragraph('自动添加的第一段文字。')
para_2=doc.add_paragraph('自动添加的第二段文字。')

para_2.add_run('继续添加第二段文字。')

doc.save(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据\word_sample_write.docx')
```

常用办公文件格式的处理：POWERPOINT

python-pptx — python-pptx 0.6.21 documentation

python-pptx库

使用*python* 操作*PPT*，需要使用的模块就是*python-pptx*，下面来对该模块做一个简单的介绍。

这里提前做一个说明：*python* 操作*PPT*，最好是我们提前设计好自己的一套样式，然后利用进行*python* 进行内容的获取和填充（最主要的功能！），最好是不用使用*python* 代码操作*PPT* 的格式，格式的修改肯定不如我们直接在*PPT*中修改方便。

✓ 可以创建、修改*PPT（.pptx）* 文件

✓ 需要单独安装，不包含在*Python*标准模块里

模块的安装与导入

1）模块的安装：***pip install python-pptx***

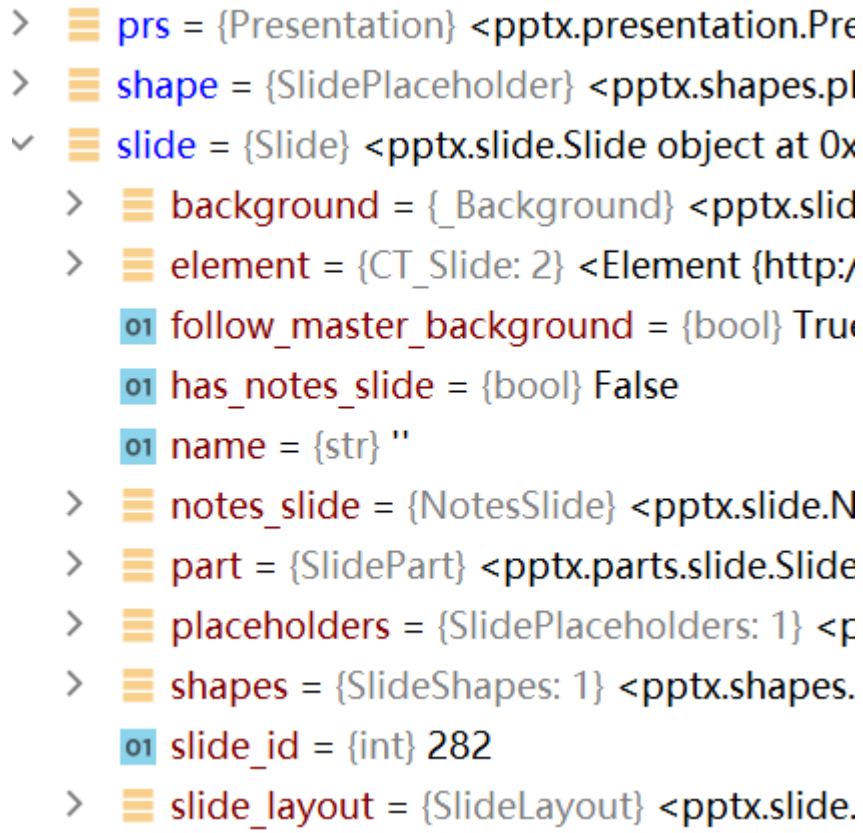
2）模块的导入：安装的库是***python-pptx***，但是导入则是***import pptx***

常用办公文件格式的处理：POWERPOINT-文档对象

PPT的文档对象

在使用*python*操作*PPT*之前，首先应该清楚*PPT*文件的结构。*python-pptx*库对*PPT*文件操作的主要对象类型包括：

- ✓ ***Presentation***：整个文稿文件
- ✓ ***Slide***：一张幻灯片
- ✓ ***SlideShapes, NotesSlide, SlidePlaceholders, Chart , Table***等：幻灯片中的形状、文本、轮廓等内容和格式方面的对象



常用办公文件格式的处理：POWERPOINT-读取

- ✓ 获取*Slide*
- ✓ 获取*Shape*形状
- ✓ 判断每个*Shape*中是否存在文字
- ✓ 获取某一页*Slide*中的内容
- ✓ 获取*Shape*中的某*Paragraph*

可以灵活有选择地获取某些Slide中的某些Shape的某些Paragraph的内容

```
from pptx import Presentation

prs = Presentation(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据\云计算网络.pptx')
for slide in prs.slides:
    print(slide)
    for shape in slide.shapes:
        print(shape)
        """注意：这里得到的Shape对象，并不能看出什么，接着往下看。"""
        if shape.has_text_frame:
            text_frame = shape.text_frame
            print(text_frame.text)

for i, slide in enumerate(prs.slides):
    if i == 5:
        for shape in slide.shapes:
            if shape.has_text_frame:
                text_frame = shape.text_frame
                for paragraph in text_frame.paragraphs:
                    print(paragraph.text)
```

常用办公文件格式的处理：POWERPOINT-写入

1）幻灯片模板及占位符的概念

2）怎么自定义母版?

3）什么是版式?
版式的设置存在*prs.slide_layouts*中，是一个序列，可以按需选取其中的某个版式。这里首先把版式内容输出记录下来备用。

4）添加*Slide*和内容
这里就需要使用上述的自定义母版。因为毕竟是使用*python*操作*PPT*，我们可以定义好自己想要展示的*PPT*母版，然后借助代码完成*PPT*的内容写入操作。

```
from pptx import Presentation

prs = Presentation(r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据\空白.pptx')
# prs.slide_layouts[]表示的是ppt中不同的版式
for layout in prs.slide_layouts:
    slide = prs.slides.add_slide(layout)
    for shape in slide.placeholders:
        phf = shape.placeholder_format
        print(f"{phf.idx}--{shape.name}--{phf.type}")
        shape.text = f"{phf.idx}--{shape.name}--{phf.type}"
# 注意：做完这个操作，一定要记得保存一下！
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\缺省模板.pptx")
```

常用办公文件格式的处理：POWERPOINT-写入

- ✓ 文本内容的填写
- ✓ 添加段落，并给段落设定层级关系
- ✓ 添加一个文本框

```
# 添加一个文本框
from pptx.util import Cm, Pt
black_slide_layout = prs.slide_layouts[0]
slide = prs.slides.add_slide(black_slide_layout)
left = top = width = height = Cm(3)
text_box = slide.shapes.add_textbox(left, top, width,
height)
tf = text_box.text_frame
tf.text = "这是一段文本框里面的文字"
p = tf.add_paragraph()
p.text = "这是第二段文字，加粗，字号40"
p.font.bold = True
p.font.size = Pt(40)
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数
据\添加一个文本框0.pptx")
```

```
# 填写文本内容
slide = prs.slides.add_slide(prs.slide_layouts[0])
name = slide.placeholders[0]
why = slide.placeholders[1]
name.text = “黄同学”
why.text = “学习很主动”
prs.save(r“C:\工作目录\北邮\北邮教学\2022 网工\示例数据\内容填充.pptx”)

# 添加段落
slide1 = prs.slides.add_slide(prs.slide_layouts[1])
shapes = slide1.shapes
title_shape = shapes.title

body_shape = shapes.placeholders[1]
title_shape.text = “这是一个标题”
tf = body_shape.text_frame   # 给body占位符添加内容
tf.text = “带圆点的符号1”

p = tf.add_paragraph()  # 在原来的基础上，添加第一个段落
p.text = “带圆点的符号2”
p.level = 1                    # 设置为层级1

p = tf.add_paragraph()  # 继续添加第二个段落
p.text = "带圆点的符号3"
p.level = 2                     # 设置为层级2

prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\嘿嘿.pptx")
```

常用办公文件格式的处理：POWERPOINT-写入

- ✓ 添加图片
- ✓ 添加表格

```
# 添加图片
prs = Presentation()
black_slide_layout = prs.slide_layouts[8]
slide = prs.slides.add_slide(black_slide_layout)
left = top = Cm(3)
height = Cm(5.5)
pic = slide.shapes.add_picture(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\geocities-web-evolve.png", left, top)
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\添加图片1.pptx")
```

```
# 添加表格
prs = Presentation()
black_slide_layout = prs.slide_layouts[6]
slide = prs.slides.add_slide(black_slide_layout)
shapes = slide.shapes
rows, cols = 5, 3
left = top = Cm(5)
width = Cm(18)
height = Cm(3)
table = shapes.add_table(rows, cols, left, top, width, height).table
table.columns[0].width = Cm(6)
table.columns[1].width = Cm(2)
table.columns[2].width = Cm(2)
table.rows[0].height = Cm(2)
data = [["姓名","性别","成绩"],["张三","男",96],["李四","女",87],["王五","女",90],["赵六","男",78]]
for row in range(rows):
    for col in range(cols):
        table.cell(row,col).text = str(data[row][col])
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\插入表格.pptx")
```

常用办公文件格式的处理：POWERPOINT-调整样式

文本框的位置调整、背景颜色调整、边框样式调整，段落对齐方式的调整、字体样式调整等

```
# 1）文本框位置的调整
from pptx.enum.text import MSO_VERTICAL_ANCHOR, MSO_AUTO_SIZE
prs = Presentation()
black_slide_layout = prs.slide_layouts[6]
slide = prs.slides.add_slide(black_slide_layout)
left = top = width = height = Cm(3)
text_box = slide.shapes.add_textbox(left, top, width, height)
tf = text_box.text_frame
tf.text = “这是一段文本框里面的文字”
tf.margin_bottom = Cm(0.1) # 下边距
tf.margin_left = 0 # 左边距
tf.vertical_anchor = MSO_VERTICAL_ANCHOR.BOTTOM # 对齐文本方式：底端对齐
tf.word_wrap = True # 框中的文字自动换行
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\文本框样式的调整.pptx")

# 2）文本框背景颜色调整
from pptx.dml.color import RGBColor
fill = text_box.fill
fill.solid()
# 使用之前一定要导入RGBColor这个库
fill.fore_color.rgb = RGBColor(247, 150, 70)
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\文本框背景色的调整.pptx")
```

```
# 3）文本框边框样式调整
line = text_box.line
line.color.rgb = RGBColor(255, 0, 0)
line.width = Cm(0.3)
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\文本框边框样式调整.pptx")

# 4）段落调整
from pptx.enum.text import PP_PARAGRAPH_ALIGNMENT
left = top = width = height = Cm(3)
text_box = slide.shapes.add_textbox(left, top, width, height)
tf = text_box.text_frame
p = tf.add_paragraph()
p.text = "这是第二段文字"
p.alignment = PP_PARAGRAPH_ALIGNMENT.LEFT
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\段落对其调整.pptx")

# 5）字体样式调整
p.font.bold = True
p.font.name = "宋体"
p.font.color.rgb = RGBColor(247, 150, 70)
p.font.size = Pt(30)
prs.save(r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\字体样式调整.pptx")
```

常用办公文件格式的处理：PDF-读取

1）***PDF***表示*Portable Document Format*，使用*.pdf*文件扩展名。
2）***PyPDF2***。用于处理*PDF*文件的模块。
安装：*pip install PyPDF2* 导入：*import PyPDF2*
3）对比
PyPDF2：模块成熟，最后一次更新在2年前， 适合页面级操作，文字提取效果较差。
PDFMiner：擅长文字抽取，目前主分支已停止维护，取而代之的是*Pdfminer.six*。
Pdfplumber：基于*pdfminer.six*的文本内容抽取工具，使用门槛更低，如支持表格提取。

Welcome to PyPDF2 — PyPDF2 documentation

```
import PyPDF2

pdf_file=open(r'C:\工作目录\北邮\北邮教学\2022网工\示例数据\pdftest.PDF', 'rb')
pdf_reader=PyPDF2.PdfFileReader(pdf_file)

print(f'This file has {pdf_reader.numPages} pages')
page_0=pdf_reader.getPage(0)
print(page_0.extractText())

pdf_file.close()

#效果一般
```

```
This file has 10 pages
1

June 7, 2019


Testimony before the U.S.
-
China Economic and Security Review Commission
```

常用办公文件格式的处理：PDF-创建、拷贝等

1）在*PyPDF2*中，与*PdfFileReader*对象相对的是*PdfFileWriter*对象，它可以创建一个新的 *PDF*文件。但*PyPDF2*不能像 *Python*可以写入纯文本文件那样，将任意文本写入*PDF*。*PyPDF2*写入*PDF*的能力，仅限于从其他*PDF*中拷贝页面、旋转页面、重叠页面和加密文件。 模块不允许直接编辑 *PDF*，必须创建一个新的 *PDF*，然后从已有的文档拷贝内容。一般方式：

1．打开一个或多个已有的 *PDF*（源 *PDF*），得到*PdfFileReader*对象。

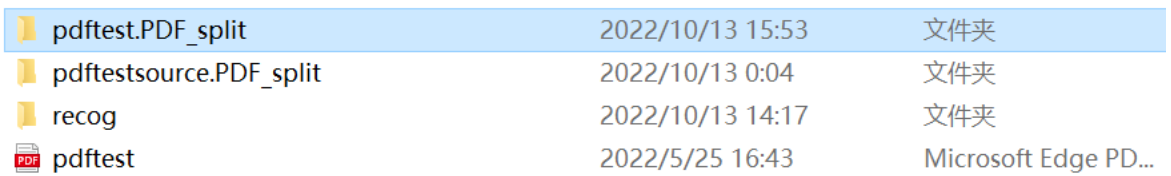
2．创建一个新的*PdfFileWriter*对象。

3．将页面从*PdfFileReader*对象拷贝到*PdfFileWriter*对象中。

4．最后，利用*PdfFileWriter*对象写入输出的*PDF*。

2）利用*PyPDF2*，从一个 *PDF*文档拷贝页面 到另一个*PDF*文档。能够组合多个*PDF*文件， 去除不想要的页面，或调整页面的次序。

3）右边的例子，把给定的pdf文件，截取指定数量的页面并分割成若干个新的*pdf*文件。




```python
from PyPDF2 import PdfFileReader, PdfFileWriter
import os

def split_pdf(filename, resultpath, numofsplits=1, start=0, end=None):
    """从filename中提取[start,end)之间的页码内容保存到resultpath目录，切分成numofsplits份"""
    # 打开原始 pdf 文件
    pdf_src = PdfFileReader(filename)
    if end is None:
        # 获取页数
        end = pdf_src.getNumPages()
    if numofsplits > end - start:
        numofsplits = end - start
    splits_size = (end - start) // numofsplits + 1
    num = start
    for i in range(numofsplits):
        with open(resultpath + f'\part_{i + 1}.pdf', "wb") as fp:
            # 创建空白pdf文件
            pdf = PdfFileWriter()
            # 提取页面内容，写入空白文件
            for j in range(splits_size):
                if num >= end: break  # 末段超出
                pdf.addPage(pdf_src.getPage(num))
                num += 1
            # 写入结果pdf
            pdf.write(fp)

src_fn = r"C:\工作目录\北邮\北邮教学\2022 网工\示例数据\pdftest.PDF"
dest_fn_path = src_fn + r'_split'
if not os.path.exists(dest_fn_path):
    os.mkdir(dest_fn_path)

split_pdf(src_fn, dest_fn_path, numofsplits=3)
```

常用办公文件格式的处理：PDF-叠加

1）*PyPDF2* 也可以将一页的内容叠加到另一页上， 这可以用来在页面上添加公司标志、时间戳或水印。利用 *Python*，很容易为多个文件添加水印，并且只针对程序指定的页面添加（比如奇数页、偶数页等）。

示例：给指定文件夹下的所有PDF 文件，加上指定的水印，并存入另外的文件夹中。其中：

- ✓ *由于处理多个文件，PdfFileWriter 对象需要能在每个文件处理完后清空才能处理下一个文件。简单的方法是删除对象后重新生成。（注意Python的GC机制）*
- ✓ *mergePage是把参数中的页面覆盖到当前页，所以水印文件的透明度需要设置为合适的值*


nder-deliver. Since its establishment in 2014, China's much-touted ser
nly spent a fraction of the $150 billion allocation and failed to spur adv
chnological frontier.[38] In robotics and smart manufacturing, local gove
uplicated projects, wasted money, and produced a glut of low-value pr

ne critical question is whether China's AI ecosystem can produce big
ndamental AI research — the cornerstone of U.S. structural advantag
ve toward inspiring these fundamental breakthroughs, the Chinese go
hinese Academy of Engineering's "Artificial Intelligence 2.0" proposal
&T megaprojects.[40] However, previous Chinese science and technolo
verted funds from high-quality labs toward more politically-connected

ndeniably, the openness of the U.S. technology ecosystem — to new
ew debates about AI ethics — provides the bedrock for its AI advantag
efense Advanced Research Projects Agency has made a variety of lo

```python
import os, PyPDF2

root_path = r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据'
src_path = root_path + r'\无水印'
dest_path = root_path + r'\有水印'
if not os.path.exists(dest_path):
    os.mkdir(dest_path)

watermark_file = open(f'{root_path}\\水印文件.pdf', 'rb')
pdf_wm_reader = PyPDF2.PdfFileReader(watermark_file)
src_pdf_list = os.listdir(src_path)
for x in src_pdf_list:
    if os.path.isdir(src_path + '\\' + x): continue
    if x.lower().endswith('.pdf'):
        pdf_writer = PyPDF2.PdfFileWriter()
        src_pdf_file = open(src_path + '\\' + x, 'rb')
        src_pdf_reader = PyPDF2.PdfFileReader(src_pdf_file)
        for page_num in range(src_pdf_reader.numPages):
            page = src_pdf_reader.getPage(page_num)
            page.mergePage(pdf_wm_reader.getPage(0))
            pdf_writer.addPage(page)

        new_pdf_file = open(dest_path + '\\' + x, 'wb')
        pdf_writer.write(new_pdf_file)
        src_pdf_file.close()
        new_pdf_file.close()
        del pdf_writer

watermark_file.close()
```

常用办公文件格式的处理：PDF-格式转换

1）还有其他第三方库比如*pdf2image*，可以把 *PDF* 文档的页面转换为图片，也是较为常见的应用。
2）右边的例子，把给定源文件目录中的*pdf* 文件的页面逐一转换为*JPEG* 文件（图形文件格式可设置）并存到给定的目标图片文件目录中。
3）其他的文件格式转换类的需求也可以找到相应的第三方库（需要时再查）。

```
    def pdf2img(self, pdf_name, img_save_dir):
        """
            将pdf文件转为jpg格式图片，若pdf有多页，则返回多张图片

            pdf_path：pdf文件路径
            img_path：转换后的图片存放路径
        """
        pages = convert_from_path(self.dir_path + '/' + pdf_name, 200)
        for i, page in enumerate(pages):
            page.save(img_save_dir + '/' + pdf_name[:-4] + '-' + str(i) + '.jpg', 'JPEG')

if __name__ == "__main__":
    dir_path = r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据\recog'
    save_dir_path = r'C:\工作目录\北邮\北邮教学\2022 网工\示例数据\img123'
    ob = IMGConverter(dir_path, save_dir_path)
    ob.run_conver()
```

```
import os
from pdf2image import convert_from_path

class IMGConverter:
    def __init__(self, dir_path, save_dir_path):
        self.dir_path = dir_path  # 保存pdf文件的文件夹路径
            self._export_folder = save_dir_path  # 保存图片的文件夹路径
            self._filename_list = list()  # 保存pdf文件名
            if not os.path.exists(self._export_folder):
            os.mkdir(self._export_folder)
        # print(self._export_folder)
        self._enumerate_filename()

    def _enumerate_filename(self):
        """
            读取所有文件名
        """
        for name in os.listdir(self.dir_path):
            if self.is_legal_postfix(name):
                self._filename_list.append(name)
            else:
                continue

    # 检查文件名是否合法
        def is_legal_postfix(self, filename):
        return filename.split('.')[-1].lower() == 'pdf' and not os.path.basename(filename).startswith('~')

    def run_conver(self):
        """
            批量处理转换操作，将一个文件夹下的pdf文件皆转换为jpg图片
        """
        print('需要转换的文件数：', len(self._filename_list))
        for filename in self._filename_list:
            print('原文件：', filename)
            self.pdf2img(filename, self._export_folder)
        print('转换完成！')
```

八 文件操作

- 文件和流
- 文件对象
- 文件的基本操作
- 随机文件读写
- 其他文件读写
- 对象序列化
- 常用办公文件格式的处理

谢谢