章节一：python 使用 openpyxl 操作 excel

1、python 怎么打开及读取表格内容？

* openpyxl 最好用的 python 操作 excel 表格库，不接受反驳；

* openpyxl 官网链接：https://openpyxl.readthedocs.io/en/stable/；

* openpyxl 只支持【.xlsx / .xlsm / .xltx / .xltm】格式的文件；

* 原文链接：https://blog.csdn.net/weixin_41261833/article/details/106028038

1）Excel 表格述语

这里需要大家仔细查看图中的每一项内容，知道什么是“行(row)、列(column)”？什么是“格子(cell)”？什么是“sheet 表”？

2）打开 Excel 表格并获取表格名称

```
from openpyxl import load_workbook
workbook = load_workbook(filename = "test.xlsx")
workbook.sheetnames
```

结果如下：

3）通过 sheet 名称获取表格

```
from openpyxl import load_workbook
workbook = load_workbook(filename = "test.xlsx")
workbook.sheetnames
sheet = workbook["Sheet1"] print(sheet)
```

结果如下：

```
In [15]: from openpyxl import load_workbook

In [16]: workbook = load_workbook(filename = "test.xlsx")
         workbook.sheetnames

Out[16]: ['Sheet1', 'Sheet2', 'Sheet3', 'Sheet4']

In [17]: sheet = workbook["Sheet1"]

In [18]: print(sheet)

         <Worksheet "Sheet1">
```

4）获取表格的尺寸大小(几行几列数据)

这里所说的尺寸大小，指的是 excel 表格中的数据有几行几列，针对的是不同的 sheet 而言。

```
sheet.dimensions
```

结果如下：

```
In [15]: from openpyxl import load_workbook

In [16]: workbook = load_workbook(filename = "test.xlsx")
         workbook.sheetnames

Out[16]: ['Sheet1', 'Sheet2', 'Sheet3', 'Sheet4']

In [17]: sheet = workbook["Sheet1"]

In [18]: print(sheet)

         <Worksheet "Sheet1">

In [19]: sheet.dimensions

Out[19]: 'A1:H11'
```

	A	B	C	D	E	F	G	H
1	name	dates	语文	数学	英语	物理	化学	生物
2	赵云	1993/8/12	33	93	39	57	46	54
3	诸葛亮	1994/6/22	37	98	49	97	91	31
4	孙权	1990/12/12	53	82	98	37	54	54
5	刘备	1993/12/2	73	70	73	78	60	92
6	貂蝉	1992/10/1	97	68	82	48	37	40
7	马超	1995/5/13	54	92	30	55	76	90
8	孟获	1993/4/10	64	36	70	84	48	36
9	费祎	1993/8/1	61	35	65	42	87	63
10	曹操	1999/6/1	59	50	36	70	56	100
11	曹丕	1993/3/28	50	45	40	86	69	40
12								

5）获取表格内某个格子的数据

① sheet["A1"]方式

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active print(sheet)
cell1 = sheet["A1"]
cell2 = sheet["C11"]
print(cell1.value, cell2.value)
"""
workbook.active 打开激活的表格；
sheet["A1"] 获取 A1 格子的数据；
cell.value 获取格子中的值；
"""
```

结果如下：

② sheet.cell(row=, column=)方式

下面这种方式更简单，大家可以对比这两种方式；

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active
print(sheet)
cell1 = sheet.cell(row = 1,column = 1)
cell2 = sheet.cell(row = 11,column = 3)
print(cell1.value, cell2.value)
```

结果如下：

```
In [32]: workbook = load_workbook(filename = "test.xlsx")
         sheet = workbook.active
         print(sheet)
         cell1 = sheet["A1"]
         cell2 = sheet["C11"]
         print(cell1.value, cell2.value)

         <Worksheet "Sheet4">
         name 190cm

In [33]: workbook = load_workbook(filename = "test.xlsx")
         sheet = workbook.active
         print(sheet)
         cell1 = sheet.cell(row = 1,column = 1)
         cell2 = sheet.cell(row = 11,column = 3)
         print(cell1.value, cell2.value)

         <Worksheet "Sheet4">
         name 190cm
```

6）获取某个格子的行数、列数、坐标

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active
print(sheet)
cell1 = sheet["A1"]
cell2 = sheet["C11"]
```

```
print(cell1.value, cell1.row, cell1.column, cell1.coordinate)
print(cell2.value, cell2.row, cell2.column, cell2.coordinate)
"""
.row 获取某个格子的行数;
.columns 获取某个格子的列数;
.corordinate 获取某个格子的坐标;
 """
```

结果如下：

```
In [27]: workbook = load_workbook(filename = "test.xlsx")
         sheet = workbook.active
         print(sheet)
         cell1 = sheet["A1"]
         cell2 = sheet["C11"]
         print(cell1.value, cell1.row, cell1.column, cell1.coordinate)
         print(cell2.value, cell2.row, cell2.column, cell2.coordinate)

<Worksheet "Sheet4">
name 1 A A1
190cm 11 C C11
```

	A	B	C
1	name	性别	身高
2	赵云	男	170cm
3	诸葛亮	1	182cm
4	小乔	0	1.6m
5	刘备	男	169cm
6	貂蝉	女	162cm
7	马超	1	1.75m
8	大乔	女	155cm
9	费祎	1	1.88m
10	曹操	男	1.72m
11	曹丕	男	190cm
12			

7）获取一系列格子

① sheet[]方式

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active print(sheet)
# 获取 A1:C2 区域的值
cell = sheet["A1:C2"]
print(cell)
for i in cell:
    for j in i:
        print(j.value)
```

结果如下：

特别的：如果我们只想获取“A 列”，或者获取“A-C 列”，可以采取如下方式：

```
sheet["A"] --- 获取 A 列的数据
sheet["A:C"] --- 获取 A,B,C 三列的数据
sheet[5] --- 只获取第 5 行的数据
```

② .iter_rows()方式

* 当然有.iter_rows()方式，肯定也会有.iter_cols()方式，只不过一个是按行读取，一个是按列读取。

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active print(sheet)
# 按行获取值
for i in sheet.iter_rows(min_row=2, max_row=5, min_col=1,
max_col=2):
    for j in i:
        print(j.value)
# 按列获取值
for i in sheet.iter_cols(min_row=2, max_row=5, min_col=1,
max_col=2):
    for j in i:
        print(j.value)
```

结果如下：

③ sheet.rows()

* 帮助我们获取所有行

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active
print(sheet)
for i in sheet.rows:
    print(i)
```

结果如下：

```
In [68]: workbook = load_workbook(filename = "test.xlsx")
         sheet = workbook.active
         print(sheet)

         for i in sheet.rows:
                 print(i)

<Worksheet "Sheet4">                    一共11行、3列数据
(<Cell 'Sheet4'.A1>, <Cell 'Sheet4'.B1>, <Cell 'Sheet4'.C1>)
(<Cell 'Sheet4'.A2>, <Cell 'Sheet4'.B2>, <Cell 'Sheet4'.C2>)
(<Cell 'Sheet4'.A3>, <Cell 'Sheet4'.B3>, <Cell 'Sheet4'.C3>)
(<Cell 'Sheet4'.A4>, <Cell 'Sheet4'.B4>, <Cell 'Sheet4'.C4>)
(<Cell 'Sheet4'.A5>, <Cell 'Sheet4'.B5>, <Cell 'Sheet4'.C5>)
(<Cell 'Sheet4'.A6>, <Cell 'Sheet4'.B6>, <Cell 'Sheet4'.C6>)
(<Cell 'Sheet4'.A7>, <Cell 'Sheet4'.B7>, <Cell 'Sheet4'.C7>)
(<Cell 'Sheet4'.A8>, <Cell 'Sheet4'.B8>, <Cell 'Sheet4'.C8>)
(<Cell 'Sheet4'.A9>, <Cell 'Sheet4'.B9>, <Cell 'Sheet4'.C9>)
(<Cell 'Sheet4'.A10>, <Cell 'Sheet4'.B10>, <Cell 'Sheet4'.C10>)
(<Cell 'Sheet4'.A11>, <Cell 'Sheet4'.B11>, <Cell 'Sheet4'.C11>)
```

2、python 如何向 excel 中写入某些内容？

1）修改表格中的内容

① 向某个格子中写入内容并保存

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active
print(sheet)
```

```
sheet["A1"] = "哈喽"
# 这句代码也可以改为 cell = sheet["A1"] cell.value = "哈喽"
workbook.save(filename = "哈喽.xlsx")
"""
注意：我们将“A1”单元格的数据改为了“哈喽”，并另存为了“哈喽.xlsx”文件。 如果我们保存的时候，不修改表名，相当于直接修改源文件；
"""
```

结果如下：

② .append()：向表格中插入行数据(很有用)

* .append()方式：会在表格已有的数据后面，增添这些数(按行插入)；

* 这个操作很有用，爬虫得到的数据，可以使用该方式保存成 Excel 文件；

```
workbook = load_workbook(filename = "test.xlsx")
```

```
sheet = workbook.active
print(sheet)
data = [
        ["唐僧","男","180cm"],
        ["孙悟空","男","188cm"],
        ["猪八戒","男","175cm"],
        ["沙僧","男","176cm"],
      ]
for row in data:
    sheet.append(row)
workbook.save(filename = "test.xlsx")
```

结果如下：

③ 在 python 中使用 excel 函数公式(很有用)

```
# 这是我们在 excel 中输入的公式
=IF(RIGHT(C2,2)="cm",C2,SUBSTITUTE(C2,"m",""))*100&"cm")
# 那么，在 python 中怎么插入 excel 公式呢？
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active
print(sheet)
sheet["D1"] = "标准身高"
for i in range(2,16):
    sheet["D{}".format(i)] =
'=IF(RIGHT(C{},2)="cm",C{},SUBSTITUTE(C{},"m",""))*100&"cm")'.format(i,i,i)
workbook.save(filename = "test.xlsx")
```

结果如下：

此时，你肯定会好奇，python 究竟支持写哪些“excel 函数公式”呢？我们可以使用如下操作查看一下。

```
import openpyxl
from openpyxl.utils import FORMULAE
print(FORMULAE)
```

结果如下：

```
In [84]: import openpyxl
         from openpyxl.utils import FORMULAE
         print(FORMULAE)
```

截取部分如下

```
frozenset({'MIN', 'FV', 'BIN2OCT', 'PEARSON', 'OCT2DEC', 'COUPDAYBS', 'EVEN', 'ACOSH', 'IMSUB', 'DGET', 'AMORLINC', 'VARP', 'INTERCEP
T', 'ASIN', 'RANK', 'OFFSET', 'RATE', 'ISBLANK', 'DSUM', 'QUOTIENT', 'COUPNCD', 'NORMINV', 'LEN', 'YIELD', 'CUBESETCOUNT', 'HLOOKUP',
'DURATION', 'MIRR', 'LOGNORMDIST', 'TRUNC', 'COUNTIFS', 'ODDLPRICE', 'AND', 'ZTEST', 'XIRR', 'IMSIN', 'ATAN2', 'ATANH', 'TIME', 'EOMONT
H', 'DEGREES', 'IMEXP', 'CHIDIST', 'COMPLEX', 'AVEDEV', 'RADIANS', 'TEXT', 'STDEVPA STEYX', 'TBILLYIELD', 'MROUND', 'DELTA', 'GAMMADIS
T', 'PRICEMAT', 'SUM', 'CRITBINOM', 'COUPPCD', 'EFFECT', 'WORKDAY.INTL', 'REPLACEB', 'FTEST', 'DEC2BIN', 'FALSE', 'GESTEP', 'MULTINOMIA
L', 'T', 'PHONETIC', 'ISERR', 'ASC', 'HOUR', 'RECEIVED', 'HYPERLINK', 'CEILING', 'ISPMT', 'RANDBETWEEN', 'CONVERT', 'CUBERANKEDMEMBER',
'INDIRECT', 'PPMT', 'WORKDAY ', 'DEVSQ', 'AMORDEGRC', 'DATE', 'INT', 'ERF', 'ERROR.TYPE', 'NA', 'SUMXMY2', 'GETPIVOTDATA', 'DOLLARDE',
'BESSELI', 'IMCONJUGATE', 'FACTDOUBLE', 'VAR', 'TREND', 'PERCENTRANK', 'MINUTE', 'IMABS', 'INTRATE', 'IMCOS', 'TRUE ADDRESS', 'CUBESE
T', 'IMREAL', 'ROMAN', 'ACOS', 'NETWORKDAYS.INTL', 'LCM', 'SQRT', 'AVERAGEA', 'SUBTOTAL', 'RAND', 'LOGINV', 'ODDFPRICE', 'MODE', 'BESSE
LK', 'N', 'HEX2OCT', 'TINV', 'RIGHT', 'OR', 'GCD', 'EDATE', 'SYD', 'ERFC', 'IMLOG2', 'IMPOWER', 'AVERAGE', 'DVARP', 'OCT2HEX', 'DSTDE
V', 'DAYS360', 'ODDFYIELD', 'NOW', 'MONTH', 'PV', 'CLEAN', 'BINOMDIST', 'AVERAGEIF', 'COLUMN', 'EXACT', 'YEARFRAC', 'IMLN', 'SUMX2PY2',
'KURT', 'TDIST', 'IMSUM', 'ODDLYIELD', 'INFO', 'ISNA', 'NORMDIST', 'EXP', 'TANH', 'GAMMALN', 'MAX', 'DISC', 'SUMIFS', 'SERIESSUM', 'SUM
X2MY2', 'LENB', 'CUBEMEMBER', 'DB', 'DMAX', 'BIN2DEC', 'TYPE', 'ASINH', 'FVSCHEDULE', 'DATEDIF', 'IMDIV', 'LARGE', 'DEC2OCT', 'FIND',
'COUNTBLANK', 'SUMSQ', 'LOOKUP', 'ACCRINT', 'HYPGEOMDIST', 'AREAS', 'CUBEKPIMEMBER', 'MINA', 'DSTDEVP', 'EXPONDIST', 'FACT', 'YIELDDIS
C', 'MOD', 'CONFIDENCE', 'ISEVEN', 'LEFT', 'SECOND', 'ABS', 'PRODUCT', 'COMBIN', 'DVAR', 'MMULT', 'PROB', 'TRIMMEAN', 'CUBEVALUE', 'SKE
W', 'ISREF', 'BESSELY', 'MIDB', 'BETAINV', 'JIS', 'ISNONTEXT', 'SUBSTITUTE', 'COVAR', 'CONCATENATE', 'ROUNDUP', 'ROUND', 'DMIN', 'YIELD
MAT', 'ACCRINTM', 'TODAY', 'MINVERSE', 'AVERAGEIFS', 'PI', 'FIXED', 'TIMEVALUE', 'IF', 'IMLOG10', 'IFERROR', 'DEC2HEX', 'GROWTH', 'SIG
N', 'NPER', 'MAXA', 'PRICEDISC', 'COUPNUM', 'NORMSINV', 'VARA', 'DPRODUCT', 'SIN', 'CHAR', 'VDB', 'LEFTB', 'LOG', 'INDEX', 'PRICE', 'NO
T', 'TRANSPOSE', 'PERMUT', 'ODD', 'VLOOKUP', 'REPLACE', 'FDIST', 'COUPDAYS', 'MDURATION', 'FISHER', 'IMPRODUCT', 'DOLLAR', 'PROPER', 'S
UMIF', 'HEX2BIN', 'WEEKNUM', 'TAN', 'IMARGUMENT', 'FINV', 'VALUE', 'BETADIST', 'ROUNDDOWN', 'CHITEST', 'NEGBINOMDIST', 'ISERROR', 'STDE
V STDEVA', 'IMAGINARY', 'LOGEST', 'CUMPRINC', 'QUARTILE', 'COUPDAYSNC', 'NPV', 'MEDIAN', 'RIGHTB', 'DOLLARFR', 'COSH', 'TTEST', 'OCT2BI
N', 'COUNTA', 'YEAR', 'SLOPE', 'GEOMEAN', 'MATCH', 'TRIM', 'PMT', 'COUNTIF', 'LOWER', 'MID', 'VARPA', 'CELL', 'FREQUENCY', 'SMALL', 'DC
OUNT', 'SEARCHB', 'ISNUMBER', 'BAHTTEXT', 'POWER', 'REPT', 'MDETERM', 'HARMEAN', 'TBILLEQ', 'ROWS', 'ROW', 'CORREL', 'LOG10', 'ISO.CEIL
ING', 'SQRTPI', 'IMSQRT', 'SLN', 'PERCENTILE', 'CUMIPMT', 'ISODD', 'STDEVP', 'COS', 'FINDB', 'RSQ', 'SUMPRODUCT', 'TBILLPRICE', 'RTD',
'NORMSDIST', 'CODE', 'XNPV', 'DCOUNTA', 'FISHERINV', 'ISLOGICAL', 'UPPER', 'ISTEXT', 'COLUMNS', 'NETWORKDAYS', 'FLOOR', 'LINEST', 'GAMM
```

④ .insert_cols()和.insert_rows()：插入空行和空列

* .insert_cols(idx=数字编号, amount=要插入的列数)，插入的位置是在 idx 列数的左侧插入；

* .insert_rows(idx=数字编号, amount=要插入的行数)，插入的行数是在 idx 行数的下方插入；

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active
print(sheet)
sheet.insert_cols(idx=4,amount=2)
sheet.insert_rows(idx=5,amount=4)
workbook.save(filename = "test.xlsx")
```

结果如下：

	A	B	C	D	E	F	G
1	姓名	性别	身高			标准身高	
2	赵云	男	170cm			170cm	
3	诸葛亮	1	182cm			182cm	
4	小乔	0	1.6m			160cm	
5							
6							
7							
8							
9	刘备	男	169cm			#VALUE!	
10	貂蝉	女	162cm			#VALUE!	
11	马超	1	1.75m			#VALUE!	
12	大乔	女	155cm			#VALUE!	
13	费祎	1	1.88m			169cm	
14	曹操	男	1.72m			162cm	
15	曹丕	男	190cm			175cm	
16						155cm	
17						188cm	
18						172cm	
19						190cm	
20							
21							
22							

⑤ .delete_rows()和.delete_cols()：删除行和列

* .delete_rows(idx=数字编号, amount=要删除的行数)

* .delete_cols(idx=数字编号, amount=要删除的列数)

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active print(sheet)
# 删除第一列，第一行
sheet.delete_cols(idx=1)
sheet.delete_rows(idx=1)
workbook.save(filename = "test.xlsx")
```

结果如下：

	A	B	C	D	E	F
1	男	170cm			#VALUE!	
2	1	182cm			#VALUE!	
3	0	1.6m			#VALUE!	
4						
5						
6						
7						
8	男	169cm			#VALUE!	
9	女	162cm			#VALUE!	
10	1	1.75m			#VALUE!	
11	女	155cm			#VALUE!	
12	1	1.88m			#VALUE!	
13	男	1.72m			#VALUE!	
14	男	190cm			#VALUE!	
15					#VALUE!	
16					#VALUE!	
17					#VALUE!	
18					#VALUE!	
19						
20						
21						
22						

⑥ .move_range()：移动格子

* .move_range("数据区域",rows=,cols=)：正整数为向下或向右、负整数为向左或向上；

向左移动两列，向下移动两行

```
sheet.move_range("C1:D4",rows=2,cols=-1)
```

演示效果如下：

	A	B	C	D
1	你好啊	123	张三	1
2	你好啊	321	李四	2
3			王五	3
4			赵六	4

	A	B
1	你好啊	123
2	你好啊	321
3	张三	1
4	李四	2
5	王五	3
6	赵六	4

⑦ .create_sheet()：创建新的 sheet 表格

* .create_sheet("新的 sheet 名")：创建一个新的 sheet 表；

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active
print(sheet)
workbook.create_sheet("我是一个新的 sheet")
print(workbook.sheetnames)
workbook.save(filename = "test.xlsx")
```

结果如下：

```
In [93]: workbook = load_workbook(filename = "test.xlsx")
         sheet = workbook.active
         print(sheet)
         workbook.create_sheet("我是一个新的sheet")
         print(workbook.sheetnames)
         workbook.save(filename = "test.xlsx")

         <Worksheet "Sheet4">
         ['Sheet1', 'Sheet2', 'Sheet3', 'Sheet4', '我是一个新的sheet']
```

⑧ .remove()：删除某个 sheet 表

* .remove("sheet 名")：删除某个 sheet 表；

```
workbook = load_workbook(filename = "test.xlsx")
sheet = workbook.active print(workbook.sheetnames)
# 这个相当于激活的这个 sheet 表，激活状态下，才可以操作；
sheet = workbook['我是一个新的 sheet']
print(sheet)
workbook.remove(sheet)
print(workbook.sheetnames)
workbook.save(filename = "test.xlsx")
```

结果如下：

```
In [105]: workbook = load_workbook(filename = "test.xlsx")
          sheet = workbook.active
          print(workbook.sheetnames)
          # 这个相当于激活的这个sheet表，激活状态下，才可以操作；
          sheet = workbook['我是一个新的sheet']
          print(sheet)
          workbook.remove(sheet)
          print(workbook.sheetnames)
          workbook.save(filename = "test.xlsx")

          ['Sheet1', 'Sheet2', 'Sheet3', 'Sheet4', '我是一个新的sheet']
          <Worksheet "我是一个新的sheet">
          ['Sheet1', 'Sheet2', 'Sheet3', 'Sheet4']
```

⑨ .copy_worksheet()：复制一个 sheet 表到另外一张 excel 表

* 这个操作的实质，就是复制某个 excel 表中的 sheet 表，然后将文件存储到另外一张 excel 表中；

```
workbook = load_workbook(filename = "a.xlsx")
sheet = workbook.active
print("a.xlsx 中有这几个 sheet 表",workbook.sheetnames)
sheet = workbook['姓名']
workbook.copy_worksheet(sheet)
workbook.save(filename = "test.xlsx")
```

结果如下：

	A	B	C	D	E	F
1	学号	姓名	年龄			
2	201901	赵1	22			
3	201902	钱1	32			
4	201903	孙1	19			
5	201904	李1	41			
6	201905	周1	28			
7						
8						
9						
10						

⑩ sheet.title：修改 sheet 表的名称

*.title = "新的 sheet 表名"

```
workbook = load_workbook(filename = "a.xlsx")
sheet = workbook.active
print(sheet)
sheet.title = "我是修改后的 sheet 名"
print(sheet)
```

结果如下：

⑪ 创建新的 excel 表格文件

```
from openpyxl import Workbook
workbook = Workbook()
sheet = workbook.active
sheet.title = "表格 1"
workbook.save(filename = "新建的 excel 表格")
```

结果如下：

⑫ sheet.freeze_panes：冻结窗口

* .freeze_panes = "单元格"

```
workbook = load_workbook(filename = "花园.xlsx")
sheet = workbook.active print(sheet) sheet.freeze_panes = "C3"
workbook.save(filename = "花园.xlsx")
"""
冻结窗口以后，你可以打开源文件，进行检验；
"""
```

结果如下：

```
In [115]: workbook = load_workbook(filename = "花园.xlsx")
          sheet = workbook.active
          print(sheet)
          sheet.freeze_panes = "C3"
          workbook.save(filename = "花园.xlsx")

          <Worksheet "Sheet1">
```

⑬ sheet.auto_filter.ref：给表格添加“筛选器”

* .auto_filter.ref = sheet.dimension 给所有字段添加筛选器；

* .auto_filter.ref = "A1" 给 A1 这个格子添加“筛选器”，就是给第一列添加“筛选器”；

```
workbook = load_workbook(filename = "花园.xlsx")
sheet = workbook.active
print(sheet)
sheet.auto_filter.ref = sheet["A1"]
workbook.save(filename = "花园.xlsx")
```

结果如下：

楼盘	楼号	单元	客户姓名	联系电话1	联系电话2
怡雅花园	D3	908	程R	32690032	
怡雅花园	A	502	陈生	33142315	
怡雅花园		704	何R	33164698	
怡雅花园	丽景楼中梯	704	何R	33164698	
怡雅花园	A座	304	李生	33224138	
怡雅花园	A座	0304	李生	33224138	
怡雅花园	7楼1	7	骆先生	33597383	
怡雅花园	7楼	7	骆先生	33597383	
怡雅花园		701	骆生	33597383	

3、批量调整字体和样式

1）修改字体样式

* Font(name=字体名称,size=字体大小,bold=是否加粗,italic=是否斜体,color=字体颜色)

```
from openpyxl.styles import Font
from openpyxl import load_workbook
workbook = load_workbook(filename="花园.xlsx")
sheet = workbook.active
cell = sheet["A1"]
font = Font(name="微软雅黑
",size=20,bold=True,italic=True,color="FF0000") cell.font = font
workbook.save(filename = "花园.xlsx")
"""
这个 color 是 RGB 的 16 进制表示，自己下去百度学习；
"""
```

结果如下：

	A	B	C	D	E	F
1	楼盘	楼号	单元	客户姓名	联系电话1	联系电话2
2	怡雅花园	D3	908	程R	32690032	
3	怡雅花园	A	502	陈生	33142315	
4	怡雅花园		704	何R	33164698	
5	怡雅花园	丽景楼中梯	704	何R	33164698	
6	怡雅花园	A座	304	李生	33224138	
7	怡雅花园	A座	0304	李生	33224138	
8	怡雅花园	7楼1	7	骆先生	33597383	
9	怡雅花园	7楼	7	骆先生	33597383	
10	怡雅花园		701	骆生	33597383	

2）获取表格中格子的字体样式

```
from openpyxl.styles import Font
from openpyxl import load_workbook
workbook = load_workbook(filename="花园.xlsx")
sheet = workbook.active
cell = sheet["A2"]
font = cell.font
print(font.name, font.size, font.bold, font.italic, font.color)
```

结果如下：

```
In [127]: from openpyxl.styles import Font
          from openpyxl import load_workbook

          workbook = load_workbook(filename="花园.xlsx")
          sheet = workbook.active
          cell = sheet["A2"]
          font = cell.font
          print(font.name, font.size, font.bold, font.italic, font.color)

          宋体 11.0 False False <openpyxl.styles.colors.Color object>
          Parameters:
          rgb=None, indexed=None, auto=None, theme=1, tint=0.0, type='theme'
```

3）设置对齐样式

* Alignment(horizontal=水平对齐模式,vertical=垂直对齐模式,text_rotation=旋转角度,wrap_text=是否自动换行)

* 水平对齐：‘distributed’，‘justify’，‘center’，‘leftfill’，‘centerContinuous’，‘right，‘general’；

* 垂直对齐：‘bottom’，‘distributed’，‘justify’，‘center’，‘top’；

```
from openpyxl.styles import Alignment
from openpyxl import load_workbook
workbook = load_workbook(filename="花园.xlsx")
sheet = workbook.active
cell = sheet["A1"]
```

```
alignment =
Alignment(horizontal="center",vertical="center",text_rotation=45,wrap_text=T
rue) cell.alignment = alignment
workbook.save(filename = "花园.xlsx")
```

结果如下：

楼盘	楼号	单元	客户姓名	联系电话1	联系电话2
怡雅花园	D3	908	程R	32690032	
怡雅花园	A	502	陈生	33142315	
怡雅花园		704	何R	33164698	
怡雅花园	丽景楼中梯	704	何R	33164698	
怡雅花园	A座	304	李生	33224138	
怡雅花园	A座	0304	李生	33224138	
怡雅花园	7楼1	7	骆先生	33597383	
怡雅花园	7楼	7	骆先生	33597383	
怡雅花园		701	骆生	33597383	

4）设置边框样式

* Side(style=边线样式，color=边线颜色)

* Border(left=左边线样式，right=右边线样式，top=上边线样式，bottom=下边线样式)

* style 参数的种类： 'double, 'mediumDashDotDot', 'slantDashDot', 'dashDotDot','dotted','hair', 'mediumDashed, 'dashed', 'dashDot', 'thin', 'mediumDashDot','medium', 'thick'

```
from openpyxl.styles import Side,Border
from openpyxl import load_workbook
workbook = load_workbook(filename="花园.xlsx")
sheet = workbook.active
```

```
cell = sheet["D6"]
side1 = Side(style="thin",color="FF0000")
side2 = Side(style="thick",color="FFFF0000")
border = Border(left=side1,right=side1,top=side2,bottom=side2)
cell.border = border
workbook.save(filename = "花园.xlsx")
```

结果如下：

	A	B	C	D	E	F
91	怡雅花园	1	302	池小姐	13660311083	
92	怡雅花园	A	0613	不祥	13668972733	
3	怡雅花园	A	502	陈生	33142315	
4	怡雅花园		704	何R	33164698	
5	怡雅花园	丽景楼中梯	704	何R	33164698	
6	怡雅花园	A座	304	李生	33224138	
7	怡雅花园	A座	0304	李生	33224138	
8	怡雅花园	7楼1	7	骆先生	33597383	
9	怡雅花园	7楼	7	骆先生	33597383	
10	怡雅花园		701	骆生	33597383	

5）设置填充样式

* PatternFill(fill_type=填充样式，fgColor=填充颜色）

* GradientFill(stop=(渐变颜色 1，渐变颜色 2……))

```
from openpyxl.styles import PatternFill,GradientFill
from openpyxl import load_workbook
workbook = load_workbook(filename="花园.xlsx")
sheet = workbook.active
cell_b9 = sheet["B9"]
pattern_fill = PatternFill(fill_type="solid",fgColor="99ccff")
cell_b9.fill = pattern_fill
```

```
cell_b10 = sheet["B10"]
gradient_fill = GradientFill(stop=("FFFFFF","99ccff","000000"))
cell_b10.fill = gradient_fill
workbook.save(filename = "花园.xlsx")
```

结果如下：

	A	B	C	D	E	F
91	怡雅花园	1	302	池小姐	13660311083	
92	怡雅花园	A	0613	不祥	13668972733	
3	怡雅花园	A	502	陈生	33142315	
4	怡雅花园		704	何R	33164698	
5	怡雅花园	丽景楼中梯	704	何R	33164698	
6	怡雅花园	A座	304	李生	33224138	
7	怡雅花园	A座	0304	李生	33224138	
8	怡雅花园	7楼1	7	骆先生	33597383	
9	怡雅花园	7楼	7	骆先生	33597383	
10	怡雅花园		701	骆生	33597383	

6）设置行高和列宽

* .row_dimensions[行编号].height = 行高

* .column_dimensions[列编号].width = 列宽

```
workbook = load_workbook(filename="花园.xlsx")
sheet = workbook.active
# 设置第 1 行的高度
sheet.row_dimensions[1].height = 50
# 设置 B 列的宽度
sheet.column_dimensions["B"].width = 20
```

```
workbook.save(filename = "花园.xlsx")
"""
sheet.row_dimensions.height = 50
sheet.column_dimensions.width = 30
这两句代码，是将整个表的行高设置为 50，列宽设置为 30；
"""
```

结果如下：

7）合并单元格

* .merge_cells(待合并的格子编号)

* .merge_cells(start_row=起始行号，start_column=起始列号，end_row=结束行号，end_column=结束列号)

```
workbook = load_workbook(filename="花园.xlsx")
```

```
sheet = workbook.active sheet.merge_cells("C1:D2")
sheet.merge_cells(start_row=7,start_column=1,end_row=8,end_column=3)
workbook.save(filename = "花园.xlsx")
```

结果如下：

当然，也有“取消合并单元格”，用法一致。

* .unmerge_cells(待合并的格子编号)

* .unmerge_cells(start_row=起始行号，start_column=起始列号，end_row=结束行号，end_column=结束列号)

章节二：python 使用 PyPDF2 和 pdfplumber 操作 pdf

1、PyPDF2 和 pdfplumber 库介绍

* PyPDF2 官网：https://pythonhosted.org/PyPDF2/
* PyPDF2 可以更好的读取、写入、分割、合并 PDF 文件；
* pdfplumber 官网：https://github.com/jsvine/pdfplumber
* pdfplumber 可以更好地读取 PDF 文件内容和提取 PDF 中的表格；
* 这两个库不属于 python 标准库，都需要单独安装；
* 原文链接：https://blog.csdn.net/weixin_41261833/article/details/106028038

2、python 提取 PDF 文字内容

1）利用 pdfplumber 提取文字

```
import PyPDF2
import pdfplumber

with pdfplumber.open("餐饮企业综合分析.pdf") as p:
    page = p.pages[2]
    print(page.extract_text())
```

结果如下：

```
In [9]: import PyPDF2
        import pdfplumber

        with pdfplumber.open("餐饮企业综合分析.pdf") as p:
            page = p.pages[2]
            print(page.extract_text())
```

```
分析餐饮企业现状与需求
餐饮企业正面临着以下几个问题。
房租价格高、人工费用高。人力成本和房租成本的上升已成为必然趋势。
□
原材料成本高。餐饮企业业务环节多，原材料种类繁多且用量不稳定，成本控制困难，现金流量大，给手
□
工操作带来了很大的难度。
服务工作效率低。采用原始点餐模式“服务员拿着一支笔和一张纸给客户点菜、下单、结账”，不能准确
□
地将客户喜欢的菜品推荐给客户，降低了客户的就餐体验，同时也增加了财务的工作量，而且速度慢，准
确率低。
大数据挖掘专家 3
```

2）利用 pdfplumber 提取表格并写入 excel

* extract_table()：如果一页有一个表格；

* extract_tables()：如果一页有多个表格；

```
import PyPDF2
import pdfplumber
from openpyxl import Workbook
with pdfplumber.open("餐饮企业综合分析.pdf") as p:
    page = p.pages[4]
    table = page.extract_table()
    print(table)
    workbook = Workbook()
```

```
    sheet = workbook.active
    for row in table:
        sheet.append(row)
    workbook.save(filename = "新pdf.xlsx")
```

结果如下：

	A	B	C	D	E	F
1		名称	含义	名称		含义
2						
3						
4		id	菜品ID	picture_file		图片文件
5		dishes_class_id	类别ID	recommend_percent		推荐度
6		dishes_name	菜品名称	weight		份量
7		price	菜品单价	taste		口味
8		amt_discount	折扣额度	creation_method		制作方法
9		sortorder	排序	description		菜品描述
10		bar_code	条码	ingredients		食材
11		cost	成本	label		标签
12		is_info_menu_item	是信息菜单项	dishes_characteristic		菜品特色
13		balance_price	抵消费用	dept_name		部门名称
14		pinyin	菜品拼音	dishes_class_name		类别名称
15		stock_count	0：已售完；1：无限量；<0：可售分量	dept_id5		部门ID

缺陷：可以看到，这里提取出来的表格有很多空行，怎么去掉这些空行呢？

判断：将列表中每个元素都连接成一个字符串，如果还是一个空字符串那么肯定就是空行。

```
import PyPDF2
import pdfplumber
from openpyxl import Workbook
with pdfplumber.open("餐饮企业综合分析.pdf") as p:
    page = p.pages[4]
    table = page.extract_table()
    print(table)
    workbook = Workbook()
    sheet = workbook.active
    for row in table:
        if not "".join([str(i) for i in row]) == "":
```

```
            sheet.append(row)
    workbook.save(filename = "新pdf.xlsx")
```

结果如下：

	A	B	C	D	E	F
1		名称	含义	名称		含义
2		id	菜品ID	picture_file		图片文件
3		dishes_class_id	类别ID	recommend_percent		推荐度
4		dishes_name	菜品名称	weight		份量
5		price	菜品单价	taste		口味
6		amt_discount	折扣额度	creation_method		制作方法
7		sortorder	排序	description		菜品描述
8		bar_code	条码	ingredients		食材
9		cost	成本	label		标签
10		is_info_menu_item	是信息菜单项	dishes_characteristic		菜品特色
11		balance_price	抵消费用	dept_name		部门名称
12		pinyin	菜品拼音	dishes_class_name		类别名称
13		stock_count	0：已售完；1：无限量；<0：可售分量	dept_id5		部门ID

3、PDF 合并及页面的排序和旋转

1）分割及合并 pdf

① 合并 pdf

首先，我们有如下几个文件，可以发现这里共有三个 PDF 文件需要我们合并。同时可以发现他们的文件名都是有规律的(如果文件名，没有先后顺序，我们合并起来就没有意义了。)

名称	修改日期	类型	大小
51-100.pdf	2020/5/3 18:08	Foxit PhantomP...	2,781 KB
101-150.pdf	2020/5/3 18:09	Foxit PhantomP...	1,855 KB
151-200.pdf	2020/5/4 6:56	Foxit PhantomP...	1,237 KB

代码如下：

```
from PyPDF2 import PdfFileReader, PdfFileWriter
pdf_writer = PdfFileWriter()
```

```
for i in range(1,len(os.listdir(r"G:\6Tipdm\7python 办公自动化
\concat_pdf"))+1):
    print(i*50+1,(i+1)*50)
    pdf_reader = PdfFileReader("G:\\6Tipdm\\7python 办公自动化\\concat_pdf\{}-
{}.pdf".format(i*50+1,(i+1)*50))
     for page in range(pdf_reader.getNumPages()):
         pdf_writer.addPage(pdf_reader.getPage(page))
with open("G:\\6Tipdm\\7python 办公自动化\\concat_pdf\merge.pdf", "wb") as
out:
    pdf_writer.write(out)
```

结果如下：

② 拆分 pdf

这里有一个“时间序列.pdf”的文件，共 3 页，我们将其每一页存为一个 PDF 文件。

代码如下：

```
from PyPDF2 import PdfFileReader, PdfFileWriter
pdf_reader = PdfFileReader(r"G:\6Tipdm\7python 办公自动化\concat_pdf\时间序列.pdf")
for page in range(pdf_reader.getNumPages()):
    pdf_writer = PdfFileWriter()
    pdf_writer.addPage(pdf_reader.getPage(page))
    with open(f"G:\\6Tipdm\\7python 办公自动化\\concat_pdf\\{page}.pdf",
        "wb") as out: pdf_writer.write(out)
```

结果如下：

2）旋转及排序 pdf

① 旋转 pdf

* .rotateClockwise(90 的倍数)：顺时针旋转 90 度

* .rotateCounterClockwise(90 的倍数)：逆时针旋转 90 度

```
from PyPDF2 import PdfFileReader, PdfFileWriter
pdf_reader = PdfFileReader(r"G:\6Tipdm\7python 办公自动化\concat_pdf\时间序列.pdf") pdf_writer = PdfFileWriter()
for page in range(pdf_reader.getNumPages()):
    if page % 2 == 0:
        rotation_page = pdf_reader.getPage(page).rotateCounterClockwise(90)
    else:
        rotation_page = pdf_reader.getPage(page).rotateClockwise(90)
    pdf_writer.addPage(rotation_page)
    with open("G:\\6Tipdm\\7python 办公自动化\\concat_pdf\\旋转.pdf", "wb")
        as out: pdf_writer.write(out)
"""
上述代码中，我们循环遍历了这个 pdf，对于偶数页我们逆时针旋转 90°，对于奇数页我们顺时针旋转 90°； 注意：旋转的角度只能是 90 的倍数；
"""
```

其中一页效果展示如下：

② 排序 pdf

需求：我们有一个 PDF 文件，我们需要倒序排列，应该怎么做呢？

首先，我们来看 python 中，怎么倒叙打印一串数字，如下图所示。

```
In [75]: for i in range(5,-1,-1):
             print(i)

         5
         4
         3
         2
         1
         0
```

那么倒序排列一个 pdf，思路同上，代码如下：

```
from PyPDF2 import PdfFileReader, PdfFileWriter
pdf_reader = PdfFileReader(r"G:\6Tipdm\7python 办公自动化\concat_pdf\时间序列.pdf") pdf_writer = PdfFileWriter()
```

```
for page in range(pdf_reader.getNumPages()-1, -1, -1):
    pdf_writer.addPage(pdf_reader.getPage(page))
with open("G:\\6Tipdm\\7python 办公自动化\\concat_pdf\\倒序.pdf", "wb") as
    out: pdf_writer.write(out)
```

结果如下：

4、pdf 批量加水印及加密、解密

1）批量加水印

```
from PyPDF2 import PdfFileReader, PdfFileWriter
from copy import copy

water = PdfFileReader(r"G:\6Tipdm\7python 办公自动化\concat_pdf\水印.pdf")
water_page = water.getPage(0)

pdf_reader = PdfFileReader(r"G:\6Tipdm\7python 办公自动化\concat_pdf\aa.pdf")
pdf_writer = PdfFileWriter()
```

```
for page in range(pdf_reader.getNumPages()):
    my_page = pdf_reader.getPage(page)
    new_page = copy(water_page)
    new_page.mergePage(my_page)
    pdf_writer.addPage(new_page)
with open("G:\\6Tipdm\\7python 办公自动化\\concat_pdf\\添加水印后的 aa.pdf",
    "wb") as out: pdf_writer.write(out)
"""
这里有一点需要注意：进行 pdf 合并的时候，我们希望“水印”在下面，文字在上面，
    因此
是“水印”.mergePage(“图片页”)
"""
```

结果如下：

2）批量加密、解密

* 这里所说的“解密”，是在知道 pdf 的密码下，去打开 pdf，而不是暴力破解；

① 加密 pdf

```
from PyPDF2 import PdfFileReader, PdfFileWriter
pdf_reader = PdfFileReader(r"G:\6Tipdm\7python 办公自动化\concat_pdf\时间序列.pdf") pdf_writer = PdfFileWriter()
for page in range(pdf_reader.getNumPages()):
    pdf_writer.addPage(pdf_reader.getPage(page))
# 添加密码
pdf_writer.encrypt("a123456")
with open("G:\\6Tipdm\\7python 办公自动化\\concat_pdf\\时间序列.pdf", "wb") as out:
    pdf_writer.write(out)
```

结果如下：

② 解密 pdf 并保存为未加密的 pdf

```
from PyPDF2 import PdfFileReader, PdfFileWriter
pdf_reader = PdfFileReader(r"G:\6Tipdm\7python 办公自动化\concat_pdf\时间序列.pdf")
# 解密
pdf pdf_reader.decrypt("a123456")
pdf_writer = PdfFileWriter()
for page in range(pdf_reader.getNumPages()):
    pdf_writer.addPage(pdf_reader.getPage(page))
with open("G:\\6Tipdm\\7python 办公自动化\\concat_pdf\\未加密的时间序列.pdf", "wb") as out:
```

```
pdf_writer.write(out)
```

结果如下：

章节三：python 使用 python-docx 操作 word

1、python-docx 库介绍

* 该模块儿可以创建、修改 Word（.docx）文件；

* 此模块儿不属于 python 标准库，需要单独安装；

* python-docx 使用官网： https://python-docx.readthedocs.io/en/latest/；

* 我们在安装此模块儿使用的是 pip install python-docx，但是在导入的时候是 import docx；

* 原文链接：https://blog.csdn.net/weixin_41261833/article/details/106028038

2、Python 读取 Word 文档内容

* 注意：每进行一个操作，必须保存一下，否则等于白做；

1）word 文档结构介绍

2）python-docx 提取文字和文字块儿

① python-docx 提取文字

有一个这样的 docx 文件，我们想要提取其中的文字，应该怎么做？

代码如下：

```
from docx import Document
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
print(doc.paragraphs)
for paragraph in doc.paragraphs:
    print(paragraph.text)
```

结果如下：

```
In [95]: from docx import Document

         doc = Document(r"G:\6Tipdm\7python办公自动化\concat_word\test1.docx")
         print(doc.paragraphs)

         for paragraph in doc.paragraphs:
             print(paragraph.text)

[<docx.text.paragraph.Paragraph object at 0x00000242CC2E4780>,
 <docx.text.paragraph.Paragraph object at 0x00000242CC2E4F98>]
这是第一段内容。
这是第二段，加粗，斜体。
```

② python-docx 提取文字块儿

```
from docx import Document
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
print(doc.paragraphs)
paragraph = doc.paragraphs[0]
runs = paragraph.runs
print(runs)
for run in paragraph.runs:
    print(run.text)
```

```
paragraph = doc.paragraphs[1]
runs = paragraph.runs
print(runs)
for run in paragraph.runs:
    print(run.text)
```

结果如下：

```
In [99]: from docx import Document

         doc = Document(r"G:\6Tipdm\7python办公自动化\concat_word\test1.docx")
         print(doc.paragraphs)
         paragraph = doc.paragraphs[0]
         runs = paragraph.runs
         print(runs)
         for run in paragraph.runs:
             print(run.text)

         [<docx.text.paragraph.Paragraph object at 0x00000242CC2E44A8>,
          <docx.text.paragraph.Paragraph object at 0x00000242CC2E4470>]
         [<docx.text.run.Run object at 0x00000242CC2E49E8>]
         这是第一段内容。

In [100]: paragraph = doc.paragraphs[1]
          runs = paragraph.runs
          print(runs)
          for run in paragraph.runs:
              print(run.text)

          [<docx.text.run.Run object at 0x00000242CC2E4780>, <docx.text.run.Run object at 0x00000242CC2E4748>,
           <docx.text.run.Run object at 0x000 00242CC2E4240>, <docx.text.run.Run object at 0x00000242CC2E4128>,
           <docx.text.run.Run object at 0x00000242CC2E4BA8>]
          这是第二段，
          加粗
          ，
          斜体
          。
```

3）利用 Python 向 Word 文档写入内容

① 添加段落

```
from docx import Document
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
# print(doc.add_heading("一级标题", level=1)) 添加一级标题的时候出错，还没有解决！ paragraph1 = doc.add_paragraph("这是一个段落")
paragraph2 = doc.add_paragraph("这是第二个段落")
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
"""
添加段落的时候，赋值给一个变量，方便我们后面进行格式调整；
"""
```

结果如下：

② 添加文字块儿

```
from docx import Document
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
# 这里相当于输入了一个空格，后面等待着文字输入
paragraph3 = doc.add_paragraph()
paragraph3.add_run("我被加粗了文字块儿").bold = True
paragraph3.add_run("，我是普通文字块儿，")
paragraph3.add_run("我是斜体文字块儿").italic = True
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
```

结果如下：

③ 添加一个分页

```
from docx import Document
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
doc.add_page_break()
```

```
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
```

结果如下：

④ 添加图片

```
from docx import Document
from docx.shared import Cm
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
doc.add_picture(r"G:\6Tipdm\7python 办公自动化
\concat_word\sun_wu_kong.png",width=Cm(5),height=Cm(5))
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
"""
Cm 模块，用于设定图片尺寸大小
"""
```

结果如下：

⑤ 添加表格

```
from docx import Document
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
list1 = [
        ["姓名","性别","家庭地址"],
        ["唐僧","男","湖北省"],
        ["孙悟空","男","北京市"],
        ["猪八戒","男","广东省"],
        ["沙和尚","男","湖南省"]
       ]
list2 = [
        ["姓名","性别","家庭地址"],
        ["貂蝉","女","河北省"],
        ["杨贵妃","女","贵州省"],
        ["西施","女","山东省"]
```

```
        ]
    table1 = doc.add_table(rows=5,cols=3)
    for row in range(5):
        cells = table1.rows[row].cells
        for col in range(3):
            cells[col].text = str(list1[row][col])
doc.add_paragraph("-----------------------------------------------------------
--") table2 = doc.add_table(rows=4,cols=3)
for row in range(4):
    cells = table2.rows[row].cells
        for col in range(3):
            cells[col].text = str(list2[row][col])
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
```

结果如下：

⑥ 提取 word 表格，并保存在 excel 中(很重要)

```
from docx import Document
from openpyxl import Workbook
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test2.docx")
t0 = doc.tables[0]
workbook = Workbook()
sheet = workbook.active
for i in range(len(t0.rows)):
    list1 = []
    for j in range(len(t0.columns)):
        list1.append(t0.cell(i,j).text)
        sheet.append(list1)
workbook.save(filename = r"G:\6Tipdm\7python 办公自动化\concat_word\来自 word 中的表.xlsx")
```

结果如下：

3、利用 Python 调整 Word 文档样式

1）修改文字字体样式

```
from docx import Document
from docx.shared import Pt,RGBColor
from docx.oxml.ns import qn
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test2.docx")
for paragraph in doc.paragraphs:
    for run in paragraph.runs:
        run.font.bold = True
        run.font.italic = True
        run.font.underline = True
        run.font.strike = True
        run.font.shadow = True
        run.font.size = Pt(18)
        run.font.color.rgb = RGBColor(255,255,0)
        run.font.name = "宋体"
        # 设置像宋体这样的中文字体，必须添加下面 2 行代码
        r = run._element.rPr.rFonts
        r.set(qn("w:eastAsia"),"宋体")
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\_test1.docx")
```

结果如下：

2）修改段落样式

① 对齐样式

```
from docx import Document
from docx.enum.text import WD_ALIGN_PARAGRAPH
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
print(doc.paragraphs[0].text)
doc.paragraphs[0].alignment = WD_ALIGN_PARAGRAPH.CENTER
# 这里设置的是居中对齐
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\对齐样式.docx")
"""
居中对齐是其中一种样式，这里还有其他选择，自己百度了解：
LEFT,CENTER,RIGHT,JUSTIFY,DISTRIBUTE,JUSTIFY_MED,JUSTIFY_HI,JUSTIFY_LOW,THAI
_JUSTIFY
"""
```

结果如下：

② 行间距调整

```
from docx import Document
from docx.enum.text import WD_ALIGN_PARAGRAPH
doc = Document(r"G:\6Tipdm\7python 办公自动化\concat_word\test1.docx")
for paragraph in doc.paragraphs:
    paragraph.paragraph_format.line_spacing = 5.0
doc.save(r"G:\6Tipdm\7python 办公自动化\concat_word\行间距.docx")
```

结果如下：

③ 段前与段后间距

* 这里提供代码，自行下去检验

段前与段后间距

paragraph.paragraph_format.space_before = Pt(12)

Pt(12)表示12磅

```
paragraph.paragraph_format.space_before = Pt(12)
paragraph.paragraph_format.space_after = Pt(12)
```

章节四：python 使用 python-pptx 操作 PPT

1、python-pptx 模块简介

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！）。

* 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
* python-pptx 官网介绍：https://python-pptx.readthedocs.io/en/latest/

2、模块的安装与导入

1）模块的安装

"Windows 用户命令行下输入" pip install python-pptx "Mac 用户命令行下输入"

```
pip3 install python-pptx
```

2）模块的导入

这里有一点需要注意的是：安装的库是 python-pptx，但是导入的时候却有点不同。

```
import pptx
```

3、python 读取 PPT 文档中的内容

1）PPT 的结构说明

在使用 python 操作 PPT 之前，首先应该清楚 PPT 的结构，这个对于之后代码的编写很有帮助。

注意：关于 run 块儿的概念，可以参考我的另外一篇文章 https://blog.csdn.net/weixin_41261833/article/details/106028038

2）获取 Slide

```
from pptx import Presentation
prs = Presentation("统计学习方法 PPT.pptx")
for slide in prs.slides:
    print(slide)
```

结果如下：

```
In [1]: import pptx

In [3]: from pptx import Presentation

        prs = Presentation("统计学习方法PPT.pptx")
        print(len(prs.slides))
        for slide in prs.slides:
            print(slide)

        20
        <pptx.slide.Slide object at 0x0000021090227228>
        <pptx.slide.Slide object at 0x0000021090227BD8>
        <pptx.slide.Slide object at 0x00000210902271D8>
        <pptx.slide.Slide object at 0x0000021090227278>
        <pptx.slide.Slide object at 0x0000021090227318>
        <pptx.slide.Slide object at 0x0000021090227368>
        <pptx.slide.Slide object at 0x0000021090227638>
        <pptx.slide.Slide object at 0x0000021090227688>
        <pptx.slide.Slide object at 0x00000210902276D8>
        <pptx.slide.Slide object at 0x00000210902277C8>
        <pptx.slide.Slide object at 0x0000021090227818>
        <pptx.slide.Slide object at 0x0000021090227868>
```

PPT共有20页，因此这里有20个slide

3）获取 Shape 形状

```
import pptx

from pptx import Presentation

prs = Presentation("统计学习方法 PPT.pptx")

for slide in prs.slides:

    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。

"""
```

结果如下：

```
In [5]: import pptx
        from pptx import Presentation

        prs = Presentation("统计学习方法PPT.pptx")
        for slide in prs.slides:
            for shape in slide.shapes:
                print(shape)

<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204EB8>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204978>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204EB8>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204978>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204EB8>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204978>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204EB8>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204978>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204EB8>
<pptx.shapes.placeholder.SlidePlaceholder object at 0x0000021090204978>
```

4）判断每个 Shape 中是否存在文字

* shape.has_text_frame ： 是否有文字

* shape.text_frame ： 获取文字框

```
import pptx

from pptx import Presentation
```

```
prs = Presentation("统计学习方法 PPT.pptx")
for slide in prs.slides:
    for shape in slide.shapes:
        if shape.has_text_frame:
            text_frame = shape.text_frame
            print(text_frame.text)
```

结果如下：

5）获取某一页 Slide 中的内容

```
import pptx
from pptx import Presentation
prs = Presentation("统计学习方法 PPT.pptx")
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
            print(text_frame.text)
```

结果如下：

6）获取 Shape 中的某个 Paragraph

```
import pptx
from pptx import Presentation
prs = Presentation("统计学习方法PPT.pptx")
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)
"""
注意：该方法和上述 4)中的方法一摸一样。上述方法是直接获取 Shpae 中的文字内容；
下面这个更灵活：先获取每个 Shape，然后在获取每个 Shape 中的 paragraph；
下面方式更好：因为我们可以针对 paragraph，写一个判断条件，只获取第几个
paragraph；
"""
```

结果如下：

```
In [29]: import pptx
         from pptx import Presentation

         prs = Presentation("统计学习方法PPT.pptx")

         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)

           一些重要的名词解释
           先验概率：是指根据以往经验和分析得到的概率，如  P(d),  P(c)。
           条件概率：一个事件发生后另一个事件发生的概率。一般的形式为  P(d|c)表示  c发生的条件下  d发生的概率。
           后验概率：事情已经发生，求这件事情发生的原因是由某个因素引起的可能性的大小。一般表示为  P(c|d)，即：执果索因。
```

4、利用 python 向 PPT 中写入内容

1）幻灯片模板及占位符的概念

2）怎么自定义母版？

https://jingyan.baidu.com/article/925f8cb8b5dfe7c0dce05671.html

3）什么是版式？

这个概念在下面的效果中，会得以体现。其中 prs.slide_layouts[]传入 0 表示获取的是第一个版式，传入 1 表示获取的是第二个版式，以此类推下去。

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。

① 占位符 id 的确认

```
import pptx
from pptx import Presentation
prs = Presentation("空白.pptx") # prs.slide_layouts[]表示的是 ppt 中不同的版式 slide = prs.slides.add_slide(prs.slide_layouts[0])
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("电子奖状模板.pptx")

"""
上述打印结果如下：
0--Title 1--TITLE (1) 这个表示标题占位符，id 为 0
13--Picture Placeholder 2--PICTURE (18) 这个表示图片占位符，id 为 13
14--Text Placeholder 3--BODY (2) 这个表示正文内容占位符，id 为 14
15--Text Placeholder 4--BODY (2) 这个表示正文内容占位符，id 为 15
我们一定要先知道每个空格的占位符 id，才可以进行下面内容的填充。
"""
```

效果如下：

② PPT 内容的填写

```
import pptx
from pptx import Presentation
```

```
prs = Presentation("空白.pptx")
# prs.slide_layouts[]表示的是 ppt 中不同的版式
slide = prs.slides.add_slide(prs.slide_layouts[0])
name = slide.placeholders[14]
why = slide.placeholders[15]
name.text = "黄同学"
why.text = "学习太积极"
prs.save("内容填充.pptx")
```

效果如下：

5）添加段落

① 占位符 id 的确认

```
import pptx
from pptx import Presentation
prs = Presentation("空白.pptx")
# prs.slide_layouts[]表示的是 ppt 中不同的版式
slide = prs.slides.add_slide(prs.slide_layouts[0])
```

```
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}"
print("-------------------------------------------")
slide = prs.slides.add_slide(prs.slide_layouts[1])
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}"
prs.save("哈哈.pptx")
```

效果如下：

```
In [62]: import pptx
         from pptx import Presentation

         prs = Presentation("finall.pptx")
         slide = prs.slides.add_slide(prs.slide_layouts[0])
         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}"
         print("-------------------------------------------")
         slide = prs.slides.add_slide(prs.slide_layouts[1])
         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}"

         prs.save("哈哈.pptx")

           0--Title 1--TITLE (1)
           13--Picture Placeholder 2--PICTURE (18)
           14--Text Placeholder 3--BODY (2)
           15--Text Placeholder 4--BODY (2)
           -------------------------------------------
           0--Title 1--TITLE (1)
           1--Content Placeholder 2--OBJECT (7)
```

② 段落的添加

```
import pptx
from pptx import Presentation
prs = Presentation("final1.pptx")
slide = prs.slides.add_slide(prs.slide_layouts[0])
name = slide.placeholders[14]
why = slide.placeholders[15]
name.text = "黄同学"
why.text = "学习太积极"
# ------------------------------------------------ #
prs1 = Presentation("final1.pptx")
slide1 = prs.slides.add_slide(prs.slide_layouts[1])
shapes = slide1.shapes
title_shape = shapes.title
# 这句代码可以改为 title_shape = shapes.placeholders[0]
body_shape = shapes.placeholders[1]
title_shape.text = "这是一个标题"

tf = body_shape.text_frame
# 这句代码就是给 body 占位符添加内容!
tf.text = "带圆点的符号 1"

p = tf.add_paragraph()
# 这个代码表示在原来的基础上，添加第一个段落!
p.text = "带圆点的符号 2"

p = tf.add_paragraph()
# 这个代码表示在原来的基础上，添加第二个段落!
```

```
p.text = "带圆点的符号 3"
prs.save("嘿嘿.pptx")
```

效果如下：

③ 给段落设定层级关系

```
import pptx
from pptx import Presentation
prs = Presentation("finall.pptx")
slide = prs.slides.add_slide(prs.slide_layouts[0])
name = slide.placeholders[14]
why = slide.placeholders[15]
name.text = "黄同学"
why.text = "学习太积极"
# ------------------------------------------------- #
prs1 = Presentation("finall.pptx")
```

```
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
tf.text = "带圆点的符号 1"

p = tf.add_paragraph()
p.text = "带圆点的符号 2"
# 原始内容的层级相当于是 0，因此这个段落我设置为层级 1，下面的段落设置为层级 2
p.level = 1

p = tf.add_paragraph()
p.text = "带圆点的符号 3"
p.level = 2
prs.save("嘻嘻.pptx")
```

效果如下：

④ 添加一个文本框

* slide.shapes.add_textbox(left, top, width, height)

```
from pptx import Presentation
from pptx.util import Cm, Pt
prs = Presentation()
# 使用第一个版式
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("添加一个文本框 0.pptx")
```

效果如下：

⑤ 添加一个图片

* slide.shapes.add_picture(图片路径, 距离左边, 距离顶端, 宽度, 高度)

第一种展示：

```
from pptx import Presentation
from pptx.util import Cm

prs = Presentation()
# 使用第七个版式
black_slide_layout = prs.slide_layouts[6]
slide = prs.slides.add_slide(black_slide_layout)

left = top = Cm(3)
pic = slide.shapes.add_picture("孙悟空.png", left, top)
prs.save("添加图片 1.pptx")
```

效果如下：

第二种展示：

```
from pptx import Presentation
from pptx.util import Cm

prs = Presentation()
# 使用第七个版式
black_slide_layout = prs.slide_layouts[6]
slide = prs.slides.add_slide(black_slide_layout)

left = top = Cm(3)
height = Cm(5.5)
pic = slide.shapes.add_picture("孙悟空.png", left, top, height=height)
prs.save("添加图片 2.pptx")
save("添加一个文本框 0.pptx")
```

效果如下：

⑥ 添加表格

* shapes.add_table(rows, cols, left, top, width, height)

```
from pptx import Presentation
from pptx.util import Cm, Pt

prs = Presentation()
# 使用第七个版式
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("插入表格.pptx")
```

结果如下：

5、PPT 文档内容样式批量调整

1）文本框位置的调整

上面我们已经知道怎么添加文本框，现在我们需要做的就是，怎么调整文本框的位置。

```
from pptx import Presentation
from pptx.util import Cm, Pt
from pptx.enum.text import MSO_ANCHOR, MSO_AUTO_SIZE

prs = Presentation()
# 使用第七个版式
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 # 下边距
# 一定要导入 MSO_ANCHOR 这个库
tf.vertical_anchor = MSO_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("文本框样式的调整.pptx")
```

结果如下：

2）文本框背景颜色调整

```
from pptx import Presentation
from pptx.util import Cm, Pt
from pptx.enum.text import MSO_ANCHOR, MSO_AUTO_SIZE
from pptx.dml.color import RGBColor
```

```
prs = Presentation()
# 使用第七个版式
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 # 下边距
tf.vertical_anchor = MSO_ANCHOR.BOTTOM
tf.word_wrap = True # 框中的文字自动换行
# ------------------------------------- #
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("文本框背景色的调整.pptx")
```

结果如下：

3）文本框边框样式调整

```
from pptx import Presentation
from pptx.util import Cm, Pt
from pptx.enum.text import MSO_ANCHOR, MSO_AUTO_SIZE
from pptx.dml.color import RGBColor

prs = Presentation()
# 使用第七个版式
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 # 下边距
tf.vertical_anchor = MSO_ANCHOR.BOTTOM
tf.word_wrap = True # 框中的文字自动换行
# ------------------------------------- #
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)
# ------------------------------------- #
line = text_box.line
line.color.rgb = RGBColor(255, 0, 0)
```

```
line.width = Cm(0.3)
prs.save("文本框边框样式调整.pptx")
```

结果如下：

4）段落对其调整

```
from pptx import Presentation
from pptx.enum.text import PP_ALIGN

prs = Presentation()
# 使用第一个版式
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
# ---------------------------- #
p = tf.add_paragraph()
p.text = "这是第二段文字"
p.alignment = PP_ALIGN.LEFT
```

```
prs.save("段落对其调整.pptx")
```

当然这里还有一些其他样式的调整，和 word 很类似，就不一一叙述了。

与word中的使用非常相似：	
.add_run()	添加新的文字块
.level	段落缩进层级
. line_spacing	段落行间距
. runs	段落内的文字块
.space_after	段后距
.space_before	段前距

5）字体样式调整

.font.name	字体名称（可直接设定中文字体）
.font.bold	是否加粗
.font.italic	是否斜体
.font.color	字体颜色
.font.size	字体大小

代码如下：

```
from pptx import Presentation
from pptx.util import Cm, Pt
from pptx.enum.text import MSO_ANCHOR, MSO_AUTO_SIZE
```

```
from pptx.dml.color import RGBColor
from pptx.enum.text import PP_ALIGN

prs = Presentation()
# 使用第七个版式
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
# ---------------------------- #
p = tf.add_paragraph()
p.text = "这是第二段文字"
p.alignment = PP_ALIGN.LEFT
# ------------------------------------ #
p.font.bold = True
p.font.name = "宋体"
p.font.color.rgb = RGBColor(247, 150, 70)
p.font.size = Pt(30)
prs.save("字体样式调整.pptx")
```

结果如下：

章节五：python 如何自动收发邮件

1、相关库介绍

1）yagmail

* Yet Another GMAIL/SMTP client；
* 非常方便的 SMTP 包，超简单的 Python 发邮件模块；
* 需要单独安装，不包含在 Python 标准模块里；
* https://github.com/kootenpv/yagmail
* 原文链接：https://blog.csdn.net/weixin_41261833/article/details/106090048

2）keyring

* 从 Python 访问系统密钥环服务(即密码不用直接写在代码里)；
* 方便、安全地储存你的密码；
* 需要单独安装，不包含在 Python 标准模块里；
* https://github.com/jaraco/keyring

3）schedule

* 超容易理解的定时任务执行器；
* 需要单独安装，不包含在 Python 标准模块里；
* https://schedule.readthedocs.io/en/stable/

4）imbox

* 简易的 Python IMAP 包；

* 进行 IMAP 相关的操作；

* 需要单独安装，不包含在 Python 标准模块里；

* https://github.com/martinrusev/imbox

5）上述库安装

```
pip install yagmail keyring schedule imbox
```

2、利用 python 发送邮件

1）邮件相关基础知识

* POP3：Post Office Protocol3 的简称，即邮局协议的第 3 个版本，它规定怎样将个人计算机连接到 Internet 的邮件服务器和下载电子邮件的电子协议。

* SMTP:Simple Mail Transfer Protocol，即简单邮件传输协议。

* IMAP:Internet Mail Access Protocol，即交互式邮件存取协议，它是跟 POP3 类似邮件访问标准协议之一。

* 注意：写代码发邮件时一定要注意不能频繁发送！容易被当做垃圾邮件被屏蔽!!!

2）python 发送邮件流程

* 以“QQ 邮箱”为例，进行说明

① 注册一个 QQ 邮箱，开通 POP3/SMTP/IMAP

* 具体步骤参考如下链接：http://xinzhi.wenda.so.com/a/1523533253610174

点击设置--》账户--》开启如下服务

注意：开启过程中，需要发送短信验证，此时会出现一个第三方密码，这个第三方密码在使用第三方软件登陆的时候，用该密码代替你的扣扣登陆密码。

② 找到 SMTP 和 IMAP 服务器的地址

如果您的邮件客户端不在上述列出的范围内，您可以尝试如下通用配置：
接收邮件服务器： imap.qq.com
发送邮件服务器： smtp.qq.com
账户名： 您的QQ邮箱账户名（如果您是VIP邮箱，账户名需要填写完整的邮件地址）
密码： 您的QQ邮箱密码
电子邮件地址： 您的QQ邮箱的完整邮件地址

③ 发送邮件之前，先使用 yagmail 存储你的邮件地址和密码

```
In [3]: import yagmail
        yagmail.register("1127421544@qq.com","[illegible]b")
```

密码

注意：这两行代码，是用于存储你的邮件地址和密码，当你执行这行代码后。你后面发送邮件的时候，就只需要显示给出你账号即可，而不用再把密码显示出来。

③ 发送第一封测试邮件

```
import yagmail
# 这里的 user 填写的是你的扣扣邮箱账号，可以看出这里并没有写我们的“密码”
yag = yagmail.SMTP(user="××××××××@qq.com",host="smtp.qq.com")
contents = ["这是第一段正文内容","这是第二段正文内容"]
# 这里填写的是你要发送的人的扣扣邮箱
yag.send("××××××××××@qq.com","这是一封邮件",contents)
"""
特别备注一下：这里可以看到，我们发送邮件的时候，此时就没有显示给出密码了。
这样做相对来说较为安全，这就是 yagmail 库的好处。
"""
```

结果如下：

④ 发送带 HTML 样式的邮件

```
import yagmail
yag = yagmail.SMTP(user="××××××××@qq.com",host="smtp.qq.com")
contents = [
'这是第一段正文内容',
'这是第二段正文内容',
'<a href="https://www.baidu.com">百度网站</a>']
yag.send("××××××××××@qq.com","这是一封邮件",contents)
```

结果如下：

⑤ 发送带附件的邮件

```
import yagmail
yag = yagmail.SMTP(user="××××××××@qq.com",host="smtp.qq.com")
contents = [
'这是第一段正文内容',
'这是第二段正文内容',
'<a href="https://www.baidu.com">百度网站</a>',
'G:\\6Tipdm\\7python 办公自动化\\我创建的压缩包.zip']
yag.send("××××××××××@qq.com","这是一封邮件",contents)
```

结果如下：

⑥ 发送带嵌入图片的邮件

```
import yagmail
yag = yagmail.SMTP(user="××××××××@qq.com",host="smtp.qq.com")
contents = [
'亲爱的××老婆',
'我爱你',
yagmail.inline('G:\\6Tipdm\\7python办公自动化\\老婆.png')]
yag.send("××××××××××@qq.com","这是一封邮件",contents)
```

⑦ 群发邮件

```
import yagmail
yag = yagmail.SMTP(user="1127421544@qq.com",host="smtp.qq.com")
contents = [
'这是第一段正文内容',
'这是第二段正文内容',
'<a href="https://www.baidu.com">百度网站</a>',
'G:\\6Tipdm\\7python办公自动化\\我创建的压缩包.zip']
# 定义一个收件人列表
shoujian_ren = ["×1@qq.com","×2@qq.com","×3@qq.com"]
yag.send(shoujian_ren,"这是一封邮件",contents)
```

⑨ 给女友定时发送邮件

利用这个功能，你可以定时给你女朋友发送邮件，表达你的关心，以免自给由于工作忙，而忽略了自己的女朋友。

```
import time
import schedule
```

```
import yagmail
import datetime
def morning():
content = ["亲爱的，记得吃早饭哦！"]
yag = yagmail.SMTP(user="1127421544@qq.com",host="smtp.qq.com")
yag.send("1127421544@qq.com","这是一封邮件",content)
def afternoon():
content = ["亲爱的，记得吃中饭哦！"]
yag = yagmail.SMTP(user="1127421544@qq.com",host="smtp.qq.com")
yag.send("1127421544@qq.com","这是一封邮件",content)
def evering():
content = ["亲爱的，记得吃晚饭哦！"]
yag = yagmail.SMTP(user="1127421544@qq.com",host="smtp.qq.com")
yag.send("1127421544@qq.com","这是一封邮件",content)
schedule.every().day.at("07:30").do(morning)
schedule.every().day.at("11:30").do(afternoon)
schedule.every().day.at("17:30").do(evering)
while True:
# 当代码完成了这一天的任务以后，自动结束任务
if datetime.datetime.now().strftime("%H:%M") == "17:31":
break
schedule.run_pending()
time.sleep(1)

"""
# 每十分钟，执行一次任务
schedule.every(10).minutes.do(job)
# 每小时，执行一次任务
schedule.every().hour.do(job)
```

```
# 每天上午十点半，执行一次任务
schedule.every().day.at("10:30").do(job)
# 每 5-10 钟，执行一次任务
schedule.every(5).to(10).minutes.do(job)
# 每周一，执行一次任务
schedule.every().monday.do(job)
# 每周三下午一点十五分，执行一次任务
schedule.every().wednesday.at("13:15").do(job)
# 每分钟的第十七秒，执行一次任务
schedule.every().minute.at(":17").do(job)
"""
```

3、利用 python 读取邮件

代码如下：

```
from imbox import Imbox
import keyring
# "1127421544@qq.com"是你的邮箱账号
pwd = keyring.get_password("yagmail", "1127421544@qq.com")
# "imap.qq.com"是你的 IMAP 邮箱服务器地址
with Imbox("imap.qq.com","1127421544@qq.com",pwd,ssl=True) as imbox:
all_inbox_messages = imbox.messages()
for uid, message in all_inbox_messages:
print(message.subject)
print(message.body["plain"])
```

如何读取未读邮件，只需要添加一个参数：

未读邮件：imbox.messages(unread=True)

```
unread_inbox_messages = imbox.messages(unread=True)
```

如果只想看红旗标记的邮件，应该怎么办：

红旗邮件：imbox.messages(flagged=True)

```
inbox_flagged_messages = imbox.messages(flagged=True)
```

如果只想看来自某个人的邮件，应该怎么做：

某收件人邮件： inbox_messages_from = imbox.messages(sent_to=邮件地址)

```
inbox_messages_to = imbox.messages(sent_to='makerbi@163.com')
```

如何按照日期筛选邮件：

按日期筛选邮件：

date__lt	某天前
date__gt	某天后
date__on	指定某一天

```
import datetime
```

```
inbox_messages_received_before = imbox.messages(date__lt=datetime.date(2019,9,18))
inbox_messages_received_after = imbox.messages(date__gt=datetime.date(2019,9,18))
inbox_messages_received_on_date = imbox.messages(date__on=datetime.date(2019,9,18))
```

设置标记已读和删除邮件：

标记已读和删除邮件

标记已读	imbox.mark_seen(uid)
删除邮件	imbox.delete(uid)

```
for uid, message in all_inbox_messages:
    if 满足某种条件的邮件:
        imbox.delete(uid)
```

章节六：python 制作电话号码归属地查询工具

1、写作目的

本文的写作目的，是基于我同学的一个业务需求，当时他领导丢给他一个表格，里面有很多电话号码，有的知道号码的归属地，有的不知道号码的归属地，然后让他将表格“归属地”这一栏补充完整。于是，我就写了这个文章。

说明：本文涉及到的一切电话号码，纯属杜撰，如果雷同，纯属雷同。

原文链接：https://blog.csdn.net/weixin_41261833/article/details/106122154

2、判断电话号码是否合法

这里我们不做太过详细、全面的判断，我们就从如下几个方面进行判断，满足如下要求，就判定该号码合法，否则就认为该号码不合法。

* ① 号码长度是否合法(大陆正常来说，号码是 11 位)；

* ② 号码是否都是数字，如果都是数字，前三位数字是否满足“移动”、“联通”、“电信”的号段；

* 某个号码同时满足上述①②要求，我们就认为该号码是合法的。

1）移动、联通、电信号段说明

你可以会有疑问，什么是“号段”？其实当你看了下面的解释后会很明白，没个电话号码前三位就属于一个号段，三大运营商，不同的 运营商有自己不同的号段，只有号段正确，才算是一个正确的电话号码。

* 联通：130，131，132，155，156，185，186，145，176

* 移动：134，135 ，136，137，138，139，147，150，151，152，157，158，159，178，182，183，184，187，188

* 电信：133,153,189

2）python 脚本

```
phone_prefix = [
'130','131','132','155','156','185','186','145','176','134','135',
'136','137','138','139','147','150','151','152','157','158','159',
'178','182','183','184','187','188','133','153','189']
def phone_check(phone_num):
if len(phone_num) != 11:
print("电话号码非法的，长度应该是 11 位!")
else:
if phone_num.isdigit():
if phone_num[:3] in phone_prefix:
print("电话号码是合法的")
else:
print("电话号码是非法的，号码前三位不是合法的号段!")
else:
print("电话号码应该全部由数字构成!")
phone_list = [
"15826829441","14445263125",
"15631243768","18677281435",
"16614256432"]
for i in phone_list:
phone_check(i)
```

结果如下：

```
In [40]: phone_prefix = ['130','131','132','155','156','185','186','145','176',
                         '134','135','136','137','138','139','147','150','151',
                         '152','157','158','159','178','182','183','184','187',
                         '188','133','153','189']
         def phone_check(phone_num):
             if len(phone_num) != 11:
                 print("电话号码非法的，长度应该是11位!")
             else:
                 if phone_num.isdigit():
                     if phone_num[:3] in phone_prefix:
                         print("电话号码是合法的")
                     else:
                         print("电话号码是非法的，号码前三位不是合法的号段!")
                 else:
                     print("电话号码应该全部由数字构成!")
```

```
In [41]: phone_list = ["15826829441","14445263125","15631243768","18677281435","16614256432"]
         for i in phone_list:
             phone_check(i)

           电话号码是合法的
           电话号码是非法的，号码前三位不是合法的号段!
           电话号码是合法的
           电话号码是合法的
           电话号码是非法的，号码前三位不是合法的号段!
```

3、电话号码的归属地查询

经过上述的判断：对于合法的号码，我们需要进行电话号码的归属地查询；对于不合法的号码，直接显示无效号码即可。

1） phone 模块的安装与导入

完成本文需求，需要安装此模块，安装方法如下。使用该模块需要特别注意的是，使用该模块进行电话号码的判断，一定要实现判断该电话号码是否合法，只有合法的电话号码，才能用于归属地查询。

```
# phone 模块的安装
pip install phone
# phone 模块的导入
from phone import Phone
```

2） python 脚本

```
from phone import Phone
def get_phone_info(phone_num):
phone_info = Phone().find(phone_num)
try: phone = phone_info['phone']
province = phone_info['province'] #所在省份
city = phone_info['city'] #所在城市
zip_code = phone_info['zip_code'] #所在城市邮编
area_code = phone_info['area_code'] #所在城市区号
phone_type = phone_info['phone_type'] #号码运营商
except:
print('无效号码')
return phone, province, city, zip_code, area_code, phone_type
phone_list = ["15826829441","15631243768","18677281435"]
for i in phone_list:
get_phone_info(i)
```

结果如下：

```
In [54]: from phone import Phone

         def get_phone_info(phone_num):
             phone_info = Phone().find(phone_num)
             try:
                 phone = phone_info['phone']
                 province = phone_info['province']    #所在省份
                 city = phone_info['city']            #所在城市
                 zip_code = phone_info['zip_code']    #所在城市邮编
                 area_code = phone_info['area_code']  #所在城市区号
                 phone_type = phone_info['phone_type'] #号码运营商
             except:
                 print('无效号码')
             return phone, province, city, zip_code, area_code, phone_type

In [55]: phone_list = ["15826829441","15631243768","18677281435"]
         for i in phone_list:
             print(get_phone_info(i))

         ('15826829441', '湖北', '孝感', '432000', '0712', '移动')
         ('15631243768', '河北', '保定', '071000', '0312', '联通')
         ('18677281435', '广西', '柳州', '545000', '0772', '联通')
```

4、案例说明

电话号码.xlsx

	A	B
1	姓名	电话号码
2	赵一	13463526412
3	貂蝉	14445263125
4	钱二	18463526412
5	孙三	15826526411
6	李四	15326526411
7	周五	17613124251
8	吴六	13451276431
9	郑七	13224213151
10	王八	15826832154
11	冯九	14531524211
12	程十	17026524211
13	张飞	16614256432

上表是我自己杜撰的一些电话号码，我们利用上述介绍的方法，先对号码进行挨个的判断，如果电话号码合法，我们再进行电话号码的归属地查询。

```
from phone import Phone
import pandas as pd
def phone_check(phone_num):
if len(phone_num) != 11:
return "电话号码非法的，长度应该是 11 位!"
else:
if phone_num.isdigit():
if phone_num[:3] in phone_prefix:
return "电话号码是合法的"
else:
return "电话号码是非法的，号码前三位不是合法的号段!"
else:
return "电话号码应该全部由数字构成!"

def get_phone_info(phone_num):
```

```
phone_info = Phone().find(phone_num)
try:
phone = phone_info['phone']
province = phone_info['province'] #所在省份
city = phone_info['city'] #所在城市
zip_code = phone_info['zip_code'] #所在城市邮编
area_code = phone_info['area_code'] #所在城市区号
phone_type = phone_info['phone_type'] #号码运营商
except:
print('无效号码')
return phone, province, city, zip_code, area_code, phone_type
phone_prefix = [
'130','131','132','155','156','185','186','145','176',
'134','135','136','137','138','139','147','150','151',
'152','157','158','159','178','182','183','184','187',
'188','133','153','189']
df = pd.read_excel(r"G:\6Tipdm\7python办公自动化\查看电话号码运营商，归属地\电话号码.xlsx")
df["电话号码"] = df["电话号码"].astype(str)
df["号码是否合法"] = df["电话号码"].apply(phone_check)
# 注意：下面这个列表解析式可能有点复杂，好好体会一下。
df["省份"] = [get_phone_info(df["电话号码"][i])[1] if element == "电话号码是合法的" else "号码无效" for i,element in enumerate(df["号码是否合法"])]
df["城市"] = [get_phone_info(df["电话号码"][i])[2] if element == "电话号码是合法的" else "号码无效" for i,element in enumerate(df["号码是否合法"])]
df["邮编"] = [get_phone_info(df["电话号码"][i])[3] if element == "电话号码是合法的" else "号码无效" for i,element in enumerate(df["号码是否合法"])]
df["区号"] = [get_phone_info(df["电话号码"][i])[4] if element == "电话号码是合法的" else "号码无效" for i,element in enumerate(df["号码是否合法"])]
```

```
df["运营商"] = [get_phone_info(df["电话号码"][i])[5] if element == "电话号码是合法的" else "号码无效" for i,element in enumerate(df["号码是否合法"])]
df
```

结果如下：

	姓名	电话号码	号码是否合法	省份	城市	邮编	区号	运营商
0	赵一	13463526412	电话号码是合法的	河北	唐山	063000	0315	移动
1	貂蝉	14445263125	电话号码是非法的，号码前三位不是合法的号段!	号码无效	号码无效	号码无效	号码无效	号码无效
2	钱二	18463526412	电话号码是合法的	山东	聊城	252000	0635	移动
3	孙三	15826526411	电话号码是合法的	湖北	荆州	434000	0716	移动
4	李四	15326526411	电话号码是合法的	黑龙江	齐齐哈尔	161000	0452	电信
5	周五	17613124251	电话号码是合法的	河南	信阳	464000	0376	联通
6	吴六	13451276431	电话号码是合法的	湖北	十堰	442000	0719	移动
7	郑七	13224213151	电话号码是合法的	辽宁	朝阳	122000	0421	联通
8	王八	15826832154	电话号码是合法的	湖北	孝感	432000	0712	移动
9	冯九	14531524211	电话号码是合法的	天津	天津	300000	022	联通
10	程十	17026524211	电话号码是非法的，号码前三位不是合法的号段!	号码无效	号码无效	号码无效	号码无效	号码无效
11	张飞	16614256432	电话号码是非法的，号码前三位不是合法的号段!	号码无效	号码无效	号码无效	号码无效	号码无效

5、展望

前几年，国家出台了“携号转网”政策，什么是“携号转网”呢？也就是说，我们如果想从移动号，变为联通号。换做以前必须扔了现在的号，重新买一个联通的号。但是现在不同了，我们可以带着这个号码，直接由移动号，变为联通号。

本文对“携号转网”的电话号码，没有判断能力，大家如果有兴趣的话，可以下去找一下“携号转网”账号的接口，看看如何能够将这个代码完善一下。