在上篇推文《操作Excel的Xlwings教程（二）》（添加链接）中重点介绍了Xlwings中的App，Book等API知识点。本次推文将继续上次的知识点，将继续介绍Xlwings中Sheet和Range等API相关的知识，并展示一些小例子，让大家可以跟着教程进行学习：

顺便说一下，本人使用的Xlwings的版本号为：0.17.1

Xwings中的Sheet对应的是Excel文件中的Sheet页，Range对应的是Excel文件的单元格，在xwlings中Range在有些时候也表示一个选定的区域。当然在选定Range之前，我们是需要进行Sheet页的引用的。

1. **Sheets**

跟Apps和App的关系一样，所有的Sheet构成Sheets集合。

假设现在我们有一个Excel文件1.xlsx，它有两个Sheet页Shee1和Shee2,我们尝试进行以下的操作:

import xlwings as xw

wb = xw.Book(**'1.xlsx'**) *#* *以xw.Book()方式打开工作簿*

print(wb.sheets) *#* *查看当前工作簿的所有Sheet详情*

print(wb.sheets.active) *#* *激活一个Sheet发现是Sheet2*

# 在Sheet1前新增一个Sheet并命名

print(wb.sheets.add(name=**'Python知识学堂'**, before=**'Sheet1'**))

# print(wb.sheets.add(name=**'Python知识学堂'**, 1))

print(wb.sheets.active) *#* *激活Sheet页，这个Sheet是新增的*

以上代码中的输出如下：

Sheets([<Sheet [1.xlsx]Sheet1>, <Sheet [1.xlsx]Sheet2>])

<Sheet [1.xlsx]Sheet2>

<Sheet [1.xlsx]Python知识学堂>

<Sheet [1.xlsx]Python知识学堂>

且打开的1.xlsx文件的Sheet页的详情如下截图：



所以我们总结出其中的两个操作:

# 返回激活(活动)工作表

wb.sheets.active

wb.sheets.add(name = None,before = None,after = None) # 新增Sheet页，参数默认均为None，其中name为名称，before指在那个页签前插入，after为哪个页签之后插入。

1. **Sheet**

之前提及到所有的Sheet构成Sheets集合，这一小节我们将介绍属于Sheet(或)Sheets的一些常用方法或属性（操作）。

假设现在我们有一个Excel文件2.xlsx，它有两个Sheet页Shee1和Shee2,我们尝试进行以下的操作:

import xlwings as xw

wb = xw.Book(**'2.xlsx'**)

wb.sheets[0]) *# 使用索引值引用"2.xlsx"文件的Sheet1*

wb.sheets[**'Sheet1'**] *# 直接使用名称引用"2.xlsx"文件的Sheet1*

wb.sheets.add() *# 新建一个Sheet。默认为Sheet3*

wb.sheets.count *# 获得工作簿中工作表sheet的数量*

**方法或属性：**

**# 我们均使用Sheet1来进行说明，sheet和Sheet不区分大小写**

sheet = wb.sheets[**'Sheet1'**] *# 引用工作表Sheet1*

sheet.active *# 获取当前活跃的工作簿*

sheet.autofit() *# 在整个工作表上自动调整宽度,可传参数*

sheet.autofit(axis=**'c'**) *# 在整个工作表上自动调整****列****的宽度*

sheet.autofit(axis=**'r'**) *# 在整个工作表上自动调整****行****的宽度*

sheet.book *# 返回指定Sheet的book,输出 <Book [2.xlsx]>*

sheet.cells *# 返回一个Range对象，该对象表示Sheet上的所有单元格（而不仅仅是当前正在使用的单元格）输出<Range [2.xlsx]Sheet1!$1:$1048576>*

sheet.name *# 获取工作表的名称*sheet.name = **'我爱python知识学堂'** *# 工作表重命名*

sheet.clear() *# 清空工作表中的内容和格式*

sheet.index *# 返回对应sheet的索引值，从0计数*

sheet.delete() *# 删除工作表*

sheet.clear\_contents() *# 清除工作表的内容，但保留格式*sheet[**'a1048576'**].end(**'up'**).row *# 工作表sheet中有数据区域最大的****行****数，法1*

sheet.used\_range.last\_cell.row *# 工作表sheet中有数据区域最大的****行****数，法2*

sheet[**'xfd1'**].end(**'left'**).column *# 工作表sheet中有数据区域最大的****列****数，法1*

sheet.used\_range.last\_cell.column *# 工作表sheet中有数据区域最大的****列****数，法2*

sheet.api.UsedRange 或sheet.used\_range *# 返回工作表中已经使用的单元格区域*

sheet.api.row(**'2:4'**).insert *# 插入行，在第2-4行插入空白行*

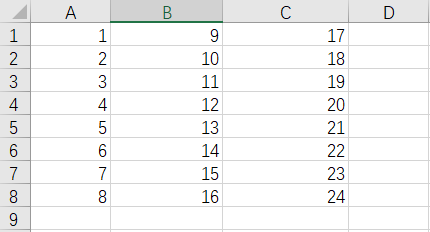
sheet.api.row(**'2:4'**).delete *# 删除行*

sheet.cells(行号,列号).value *# 取值*

1. **Range**

在Excel中我们做的中最多的就是对Excel单元格的操作，在Xlwings中涉及到的Range的方法也是比较多的。这一章我们将使用小例子的方式探究xlwings中涉及到的有关Range的方法或属性。

假设我们有一个名为“3.xlsx”的文件，其数据如下：



先导入模块，并打开工作簿，引用sheet1工作表：

import xlwings as xw

app = xw.App(visible=True,add\_book=False)  
wb = app.books.open(**'3.xlsx'**)

sheet = wb.sheets[**'Sheet1'**] *# 或wb.sheets[0]*

wb.close()  
app.quit()  
app.kill()

正如之前讲述的，上述代码是创建应用打开Excel的，这种方式打开之后需要进行关闭

以下我们使用Xlwings中的Book

import xlwings as xw

wb = xw.Book(**'3.xlsx'**)

*# sheet = wb.sheets[0]*

*# 这里我们直接引用“当前活动工作表的单元格”*

Range = xw.Range(**'A1'**)

**方法或属性：**

xw.Range(**'A1'**) *# 引用A1单元格*

xw.Range(**'A1'**).value *# 取A1单元格的值,输出1.0*

xw.Range(**'A1'**).value = 12 *# 设置值*

xw.Range(**'A1'**).raw\_value *# 直接获取并设置所使用（pywin32）引擎发送/接受的值，而无需进行任何xlwings数据清理/转换。*

xw.Range (**'A1:B2'**).value *# 引用区域并取值，输出[[1.0, 9.0], [2.0, 10.0]]，以二元list形式*

xw.Range ((1,1), (2,2)).value *# 与上述等效*

xw.Range ((**'A1'**), (**'B2'**)).value *# 与上述等效*

xw.Range (**'A9'**).add\_hyperlink(address=**'www.baidu.com'**) *# 添加超链接*

xw.Range (**'A10'**).address *# 返回表示范围参考的字符串值，输出 $A$10*

xw.Range (**'A1'**).api *# 返回所使用引擎的本机对象*

xw.Range (**'A1'**).autofit() *# 自动调整范围内所有单元格的宽度和高度。*

*# 如果仅自动调整列的宽度，使用 sheet.range('A1:B2').columns.autofit()*

*# 如果仅自动调整行的高度，使用 sheet.range('A1:B2').rows.autofit()*

xw.Range (**'A1'**).clear() *# 清除所选择单元格的内容和格式，可选择范围*

xw.Range (**'A1'**).clear\_contents() *# 清除范围的内容，但保留格式。*

xw.Range (**'A1'**).color *#获取A1单元格的背景色。*

xw.Range (**'A1'**).color = (255,255,255) *#设置A1单元格的背景色，RGB颜色*

xw.Range (**'A1'**).color = None *# 删除背景色*

xw.Range (**'B1:C4'**).column *# 返回所选范围第一列的列标，此处输出2*

xw.Range (**'B2:C4'**).row *# 返回所选范围第一行的行标，此处输出2*

xw.Range (**'A1:B2'**).count *# 返回所选范围单元格数量，此处输出4*

xw.Range (**'A1'**).current\_region *# 此属性返回一个Range对象，该对象表示由（但不包括）空白行和空白列或工作表的边缘的任意组合限制的范围，好比是一片连接的区域*

xw.Range (**'A1'**).delete() *# 删除单元格A1,有参数left和up，如delete('up')。如果省略，Excel将根据范围的形状进行决定。*

xw.Range (**'A1'**).end(**'down'**) *# 返回一个Range对象，该对象表示包含源范围的区域末尾的单元格。此处输出<Range [3.xlsx]Sheet1!$A$8>，参数可传down,up,left,right，其实也是返回ctrl + 方向*

xw.Range (**'A9'**).formula=**'=SUM(B1:B5)'** *# 设置A9单元格公式计算的值*

print(xw.Range (**'A9'**).formula) *# 输出公式值，输出'=SUM(B1:B5)'*

xw.Range (**'A1:B2'**).get\_address() *# 以指定的格式返回范围的地址*

*#参数：*

*#row\_absolute(bool ,默认为True)–设置为True可以将引用的行部分作为绝对引用返回。#column\_absolute(bool,默认为True)–设置为True可以将引用的列部分作为绝对引用返回。*

*#include\_sheetname(bool ,默认为False)–设置为True可以在地址中包含工作表名称。*

*#external(bool ,默认为False)–设置为True以返回带有工作簿和工作表名称的外部引用。*

*#具体的情况大家可以传入几个参数试试*

xw.Range (**'A1:B2'**).height *# 返回单元格(范围)的高度*

xw.Range (**'A1:B2'**).width *# 返回范围的宽度*

xw.Range (**'B2:C4'**).row\_height *# 获取范围的高度（以磅为单位*xw.Range (**'B2:C4'**).row\_height = 15 *# 设置范围的高度（以磅为单位*

*# 获取范围行和范围列*

xw.Range (**'B2:C4'**).rows *# 返回一个RangeRows对象，该对象表示指定范围内的行。*

xw.Range (**'B2:C4'**).columns *# 返回一个RangeRows对象，该对象表示指定范围内的列。*

xw.Range(**'B2:C4'**).rows.count *# 获取范围行*

xw.Range(**'B2:C4'**).columns.count *# 获取范围列*

xw.Range (**'B2'**).left *# 返回从A列的左边缘到范围左边缘的距离*

xw.Range (**'B2'**).top *# 返回从第1行的顶部边缘到范围的顶部边缘的距离*

xw.Range (**'A1'**).hyperlink *# 返回单元格中的超链接（对多个单元格没效）*

xw.Range (**'A3'**).insert(shift=**'right'**) *# 插入单元格*

range = xw.Range (**'A1:C4'**).last\_cell *#返回指定范围的右下角单元格*

range.last\_cell.row *# 4*

range.last\_cell.column *# 3*

xw.Range(**"A4:C4"**).api.merge() *# 合并单元格通过pywin32的api调用merge*

xw.Range(**"A4:C4"**).api.unmerge() *# 拆分单元格*

xw.Range(**'A1'**).number\_format *# 获取设置Range的number\_format*

xw.Range(**'A1:C3'**).number\_format = **'0.00%'** *# 设置Range的number\_format*

xw.Range(**'A1:A3'**).paste() *# 将剪贴板中的范围粘贴到指定范围*

xw.Range(**'A1:A3'**).resize(row\_size = None,column\_size = None ) *# 调整指定范围的大小*

xw.Range(**'B2:C4'**).offset(row\_offset=0,column\_offset=0) *# 选定单元格进行移动  
#row\_offset行偏移，column\_offset列偏移*

xw.Range(**'B2:C4'**).shape *# 以数组的形式返回所选范围的值*

xw.Range(**'B2:C4'**).sheet *# 返回Range所属的Sheet对象*

xw.Range(**'B2:C4'**).size *# 返回所选范围单元格个数(元素个数)*

这里想到一个参数：在读值的时候如果想将行或列方向信息需要保留下来，需要设置options的参数值，请分析这两行代码的输出：

xw.Range(**'B2:C4'**).options(ndim=2).value

xw.Range(**'A1:C1'**).options(ndim=2).value

1. **小结**

大家可以看出，这些API的知识还是很丰富的，这也是Xlwings强大的地方，当然了这些小操作结合起来就会有意想不到的效果。介绍了这么多的API知识，相必你也跃跃欲试了，赶快操作起来吧！

哦，对了，感谢大家近期的关注，由于时间的原因，目前只能一周更新一篇原创推文，文中有什么不对的地方还请大家在公众号后台留言哈！有什么好的建议也欢迎提出。

预告：下期我们将继续讲解其他相关的API知识(或者说xlwings中对象知识)。