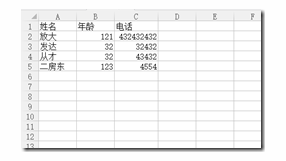
用python来自动生成excel数据文件。python处理excel文件主要是第三方模块库xlrd、xlwt、xluntils和pyExcelerator，除此之外，python处理excel还可以用win32com和openpyxl模块。

**方法一：**

小罗问我怎么从excel中读取数据，然后我百了一番，做下记录

excel数据图(小罗说数据要给客户保密，我随手写了几行数据)：



**python读取excel文件代码：**

[?](http://www.jb51.net/article/77626.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  # 读取excel数据  # 小罗的需求，取第二行以下的数据，然后取每行前13列的数据  import xlrd  data = xlrd.open\_workbook('test.xls') # 打开xls文件  table = data.sheets()[0] # 打开第一张表  nrows = table.nrows # 获取表的行数  for i in range(nrows): # 循环逐行打印  if i == 0: # 跳过第一行  continue  print table.row\_values(i)[:13] # 取前十三列 |

excel的写操作等后面用到的时候在做记录

**方法二：**

使用xlrd读取文件，使用xlwt生成Excel文件（可以控制Excel中单元格的格式）。但是用xlrd读取excel是不能对其进行操作的；而xlwt生成excel文件是不能在已有的excel文件基础上进行修改的，如需要修改文件就要使用xluntils模块。pyExcelerator模块与xlwt类似，也可以用来生成excel文件。

**1. [代码]test\_xlrd.py**

[?](http://www.jb51.net/article/77626.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | #coding=utf-8  #######################################################  #filename:test\_xlrd.py  #author:defias  #date:xxxx-xx-xx  #function：读excel文件中的数据  #######################################################  import xlrd  #打开一个workbook  workbook = xlrd.open\_workbook('E:\\Code\\Python\\testdata.xls')  #抓取所有sheet页的名称  worksheets = workbook.sheet\_names()  print('worksheets is %s' %worksheets)  #定位到sheet1  worksheet1 = workbook.sheet\_by\_name(u'Sheet1')  """  #通过索引顺序获取  worksheet1 = workbook.sheets()[0]  #或  worksheet1 = workbook.sheet\_by\_index(0)  """  """  #遍历所有sheet对象  for worksheet\_name in worksheets:  worksheet = workbook.sheet\_by\_name(worksheet\_name)  """  #遍历sheet1中所有行row  num\_rows = worksheet1.nrows  for curr\_row in range(num\_rows):  row = worksheet1.row\_values(curr\_row)  print('row%s is %s' %(curr\_row,row))  #遍历sheet1中所有列col  num\_cols = worksheet1.ncols  for curr\_col in range(num\_cols):  col = worksheet1.col\_values(curr\_col)  print('col%s is %s' %(curr\_col,col))  #遍历sheet1中所有单元格cell  for rown in range(num\_rows):  for coln in range(num\_cols):  cell = worksheet1.cell\_value(rown,coln)  print cell  """  #其他写法：  cell = worksheet1.cell(rown,coln).value  print cell  #或  cell = worksheet1.row(rown)[coln].value  print cell  #或  cell = worksheet1.col(coln)[rown].value  print cell  #获取单元格中值的类型，类型 0 empty,1 string, 2 number, 3 date, 4 boolean, 5 error  cell\_type = worksheet1.cell\_type(rown,coln)  print cell\_type  """ |

**2. [代码]test\_xlwt.py**

[?](http://www.jb51.net/article/77626.htm)

|  |  |
| --- | --- |
| 10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #coding=utf-8  #######################################################  #filename:test\_xlwt.py  #author:defias  #date:xxxx-xx-xx  #function：新建excel文件并写入数据  #######################################################  import xlwt  #创建workbook和sheet对象  workbook = xlwt.Workbook() #注意Workbook的开头W要大写  sheet1 = workbook.add\_sheet('sheet1',cell\_overwrite\_ok=True)  sheet2 = workbook.add\_sheet('sheet2',cell\_overwrite\_ok=True)  #向sheet页中写入数据  sheet1.write(0,0,'this should overwrite1')  sheet1.write(0,1,'aaaaaaaaaaaa')  sheet2.write(0,0,'this should overwrite2')  sheet2.write(1,2,'bbbbbbbbbbbbb')  """  #-----------使用样式-----------------------------------  #初始化样式  style = xlwt.XFStyle()  #为样式创建字体  font = xlwt.Font()  font.name = 'Times New Roman'  font.bold = True  #设置样式的字体  style.font = font  #使用样式  sheet.write(0,1,'some bold Times text',style)  """  #保存该excel文件,有同名文件时直接覆盖  workbook.save('E:\\Code\\Python\\test2.xls')  print '创建excel文件完成！' |

**3. [代码]test\_xlutils.py**

[?](http://www.jb51.net/article/77626.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #coding=utf-8  #######################################################  #filename:test\_xlutils.py  #author:defias  #date:xxxx-xx-xx  #function：向excel文件中写入数据  #######################################################  import xlrd  **from xlutils.copy import copy**  #打开一个workbook  rb = xlrd.open\_workbook('E:\\Code\\Python\\test1.xls')  wb = copy(rb)  #获取sheet对象，通过sheet\_by\_index()获取的sheet对象没有write()方法  ws = wb.get\_sheet(0)  #写入数据  ws.write(1, 1, 'changed!')  #添加sheet页  wb.add\_sheet('sheetnnn2',cell\_overwrite\_ok=True)  #利用保存时同名覆盖达到修改excel文件的目的,注意未被修改的内容保持不变  wb.save('E:\\Code\\Python\\test1.xls') |

**4. [代码]test\_pyExcelerator\_read.py**

[?](http://www.jb51.net/article/77626.htm)

|  |  |
| --- | --- |
| 4  5  6  7  8  9  10  11  12 | #coding=utf-8  #######################################################  #filename:test\_pyExcelerator\_read.py  #author:defias  #date:xxxx-xx-xx  #function：读excel文件中的数据  #######################################################  import pyExcelerator  #parse\_xls返回一个列表，每项都是一个sheet页的数据。  #每项是一个二元组(表名,单元格数据)。其中单元格数据为一个字典，键值就是单元格的索引(i,j)。如果某个单元格无数据，那么就不存在这个值  sheets = pyExcelerator.parse\_xls('E:\\Code\\Python\\testdata.xls')  print sheets |

**5. [代码]test\_pyExcelerator.py**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #coding=utf-8  #######################################################  #filename:test\_pyExcelerator.py  #author:defias  #date:xxxx-xx-xx  #function：新建excel文件并写入数据  #######################################################  import pyExcelerator  #创建workbook和sheet对象  wb = pyExcelerator.Workbook()  ws = wb.add\_sheet(u'第一页')  #设置样式  myfont = pyExcelerator.Font()  myfont.name = u'Times New Roman'  myfont.bold = True  mystyle = pyExcelerator.XFStyle()  mystyle.font = myfont  #写入数据，使用样式  ws.write(0,0,u'ni hao 帕索！',mystyle)  #保存该excel文件,有同名文件时直接覆盖  wb.save('E:\\Code\\Python\\mini.xls')  print '创建excel文件完成！' |

**第二part**

**python读excel——xlrd**

这个过程有几个比较麻烦的问题，比如读取日期、读合并单元格内容。下面先看看基本的操作：

首先读一个excel文件，有两个sheet，测试用第二个sheet，sheet2内容如下：

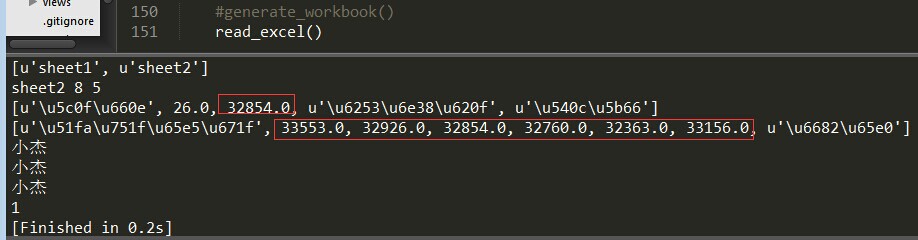


python 对 excel基本的操作如下：

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | # -\*- coding: utf-8 -\*-  import xlrd  import xlwt  from datetime import date,datetime    def read\_excel():    # 打开文件    workbook = xlrd.open\_workbook(r'F:\demo.xlsx')    # 获取所有sheet    print workbook.sheet\_names() # [u'sheet1', u'sheet2']    sheet2\_name = workbook.sheet\_names()[1]      # 根据sheet索引或者名称获取sheet内容    sheet2 = workbook.sheet\_by\_index(1) # sheet索引从0开始    sheet2 = workbook.sheet\_by\_name('sheet2')      # sheet的名称，行数，列数    print sheet2.name,sheet2.nrows,sheet2.ncols      # 获取整行和整列的值（数组）    rows = sheet2.row\_values(3) # 获取第四行内容    cols = sheet2.col\_values(2) # 获取第三列内容    print rows    print cols      # 获取单元格内容    print sheet2.cell(1,0).value.encode('utf-8')    print sheet2.cell\_value(1,0).encode('utf-8')    print sheet2.row(1)[0].value.encode('utf-8')      # 获取单元格内容的数据类型    print sheet2.cell(1,0).ctype    if \_\_name\_\_ == '\_\_main\_\_':    read\_excel() |

运行结果如下：

[](http://files.jb51.net/file_images/article/201501/2015129105924806.jpg?20150291108)

那么问题来了，上面的运行结果中红框框中的字段明明是出生日期，可显示的确实浮点数。好的，来解决第一个问题：

**1、python读取excel中单元格内容为日期的方式**

python读取excel中单元格的内容返回的有5种类型，即上面例子中的ctype:

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1 | ctype : 0 empty,1 string, 2 number, 3 date, 4 boolean, 5 error |

即date的ctype=3，这时需要使用xlrd的xldate\_as\_tuple来处理为date格式，先判断表格的ctype=3时xldate才能开始操作。现在命令行看下：

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | >>> sheet2.cell(2,2).ctype #1990/2/22  >>> sheet2.cell(2,1).ctype #24  >>> sheet2.cell(2,0).ctype #小胖  >>> sheet2.cell(2,4).ctype #空值（这里是合并单元格的原因）  >>> sheet2.cell(2,2).value #1990/2/22  33656.0  >>> xlrd.xldate\_as\_tuple(sheet2.cell\_value(2,2),workbook.datemode)  (1992, 2, 22, 0, 0, 0)  >>> date\_value = xlrd.xldate\_as\_tuple(sheet2.cell\_value(2,2),workbook.datemode)  >>> date\_value  (1992, 2, 22, 0, 0, 0)  >>> date(\*date\_value[:3])  datetime.date(1992, 2, 22)  >>> date(\*date\_value[:3]).strftime('%Y/%m/%d')  '1992/02/22' |

即可以做下简单处理，判断ctype是否等于3，如果等于3，则用时间格式处理：

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3 | if (sheet.cell(row,col).ctype == 3):    date\_value = xlrd.xldate\_as\_tuple(sheet.cell\_value(rows,3),book.datemode)    date\_tmp = date(\*date\_value[:3]).strftime('%Y/%m/%d') |

那么问题又来了，上面 sheet2.cell(2,4).ctype 返回的值是0，说明这个单元格的值是空值，明明是合并的单元格内容"好朋友"，这个是我觉得这个包功能不完善的地方，如果是合并的单元格那么应该合并的单元格的内容一样，但是它只是合并的第一个单元格的有值，其它的为空。

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | >>> sheet2.col\_values(4)  [u'\u5173\u7cfb', u'\u597d\u670b\u53cb', '', u'\u540c\u5b66', '', '', u'\u4e00\u4e2a\u4eba', '']  >>> for i in range(sheet2.nrows):    print sheet2.col\_values(4)[i]  关系  好朋友    同学      一个人    >>> sheet2.row\_values(7)  [u'\u65e0\u540d', 20.0, u'\u6682\u65e0', '', '']  >>> for i in range(sheet2.ncols):    print sheet2.row\_values(7)[i]  无名  20.0  暂无      >>> |

**2、读取合并单元格的内容**

这个是真没技巧，只能获取合并单元格的第一个cell的行列索引，才能读到值，读错了就是空值。

即合并行单元格读取行的第一个索引，合并列单元格读取列的第一个索引，如上述，读取行合并单元格"好朋友"和读取列合并单元格"暂无"只能如下方式：

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | >>> print sheet2.col\_values(4)[1]  好朋友  >>> print sheet2.row\_values(7)[2]  暂无    　>>> sheet2.merged\_cells # 明明有合并的单元格，为何这里是空   [] |

疑问又来了，合并单元格可能出现空值，但是表格本身的普通单元格也可能是空值，要怎么获取单元格所谓的"第一个行或列的索引"呢？

这就要先知道哪些是单元格是被合并的！

**3、获取合并的单元格**

读取文件的时候需要将formatting\_info参数设置为True，默认是False，所以上面获取合并的单元格数组为空，

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4 | >>> workbook = xlrd.open\_workbook(r'F:\demo.xlsx',formatting\_info=True)  >>> sheet2 = workbook.sheet\_by\_name('sheet2')  >>> sheet2.merged\_cells  [(7, 8, 2, 5), (1, 3, 4, 5), (3, 6, 4, 5)] |

merged\_cells返回的这四个参数的含义是：(row,row\_range,col,col\_range),其中[row,row\_range)包括row,不包括row\_range,col也是一样，即(1, 3, 4, 5)的含义是：第1到2行（不包括3）合并，(7, 8, 2, 5)的含义是：第2到4列合并。

利用这个，可以分别获取合并的三个单元格的内容：

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> print sheet2.cell\_value(1,4)  #(1, 3, 4, 5)  好朋友  >>> print sheet2.cell\_value(3,4)  #(3, 6, 4, 5)  同学  >>> print sheet2.cell\_value(7,2)  #(7, 8, 2, 5)  暂无 |

发现规律了没？是的，获取merge\_cells返回的row和col低位的索引即可！ 于是可以这样一劳永逸：

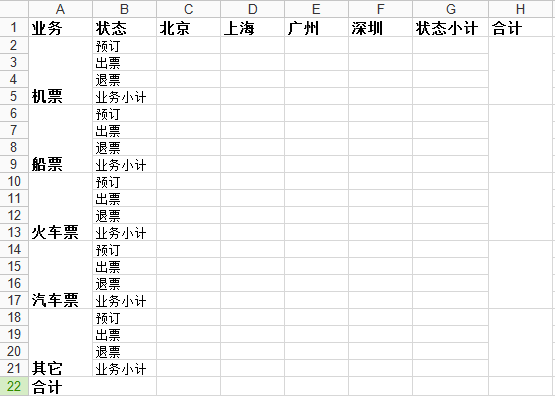
[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | >>> merge = []  >>> for (rlow,rhigh,clow,chigh) in sheet2.merged\_cells:    merge.append([rlow,clow])    >>> merge  [[7, 2], [1, 4], [3, 4]]  >>> for index in merge:    print sheet2.cell\_value(index[0],index[1])    暂无  好朋友  同学  >>> |

**python写excel——xlwt**

写excel的难点可能不在构造一个workbook的本身，而是填充的数据，不过这不在范围内。在写excel的操作中也有棘手的问题，比如写入合并的单元格就是比较麻烦的，另外写入还有不同的样式。这些要看源码才能研究的透。

我"构思"了如下面的sheet1，即要用xlwt实现的东西：



基本上看起来还算复杂，而且看起来"很正规"，完全是个人杜撰。

代码如下：

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | '''  设置单元格样式  '''    def set\_style(name,height,bold=False):    style = xlwt.XFStyle() # 初始化样式      font = xlwt.Font() # 为样式创建字体    font.name = name # 'Times New Roman'    font.bold = bold    font.color\_index = 4    font.height = height      # borders= xlwt.Borders()    # borders.left= 6    # borders.right= 6    # borders.top= 6    # borders.bottom= 6      style.font = font    # style.borders = borders      return style      #写excel  def write\_excel():    f = xlwt.Workbook() #创建工作簿      '''    创建第一个sheet:      sheet1    '''    sheet1 = f.add\_sheet(u'sheet1',cell\_overwrite\_ok=True) #创建sheet    row0 = [u'业务',u'状态',u'北京',u'上海',u'广州',u'深圳',u'状态小计',u'合计']    column0 = [u'机票',u'船票',u'火车票',u'汽车票',u'其它']    status = [u'预订',u'出票',u'退票',u'业务小计']      #生成第一行    for i in range(0,len(row0)):      sheet1.write(0,i,row0[i],set\_style('Times New Roman',220,True))      #生成第一列和最后一列(合并4行)    i, j = 1, 0    while i < 4\*len(column0) and j < len(column0):      sheet1.write\_merge(i,i+3,0,0,column0[j],set\_style('Arial',220,True)) #第一列      sheet1.write\_merge(i,i+3,7,7) #最后一列"合计"      i += 4      j += 1      sheet1.write\_merge(21,21,0,1,u'合计',set\_style('Times New Roman',220,True))      #生成第二列    i = 0    while i < 4\*len(column0):      for j in range(0,len(status)):        sheet1.write(j+i+1,1,status[j])      i += 4      f.save('demo1.xlsx') #保存文件    if \_\_name\_\_ == '\_\_main\_\_':    #generate\_workbook()    #read\_excel()    write\_excel() |

需要稍作解释的就是write\_merge方法：

write\_merge(x, x + m, y, w + n, string, sytle)  
x表示行，y表示列，m表示跨行个数，n表示跨列个数，string表示要写入的单元格内容，style表示单元格样式。其中，x，y，w，h，都是以0开始计算的。

这个和xlrd中的读合并单元格的不太一样。

如上述：sheet1.write\_merge(21,21,0,1,u'合计',set\_style('Times New Roman',220,True))

即在22行合并第1,2列，合并后的单元格内容为"合计"，并设置了style。

如果需要创建多个sheet，则只要f.add\_sheet即可。

如在上述write\_excel函数里f.save('demo1.xlsx') 这句之前再创建一个sheet2，效果如下：



代码也是真真的easy的了：

[?](http://www.jb51.net/article/60510.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | '''    创建第二个sheet:      sheet2    '''    sheet2 = f.add\_sheet(u'sheet2',cell\_overwrite\_ok=True) #创建sheet2    row0 = [u'姓名',u'年龄',u'出生日期',u'爱好',u'关系']    column0 = [u'小杰',u'小胖',u'小明',u'大神',u'大仙',u'小敏',u'无名']      #生成第一行    for i in range(0,len(row0)):      sheet2.write(0,i,row0[i],set\_style('Times New Roman',220,True))      #生成第一列    for i in range(0,len(column0)):      sheet2.write(i+1,0,column0[i],set\_style('Times New Roman',220))      sheet2.write(1,2,'1991/11/11')    sheet2.write\_merge(7,7,2,4,u'暂无') #合并列单元格    sheet2.write\_merge(1,2,4,4,u'好朋友') #合并行单元格      f.save('demo1.xlsx') #保存文件 |