

# Excel文件介绍

- 1 excel文件格式有2种，一种是xls，一种是xlsx
- 2 XLS格式是Excel 2003及之前版本的工作簿文件格式，而XLSX格式是Excel 2007及以后版本的工作簿文件格式，它们之间不兼容。XLSX格式支持更大的文件容量、更强大的功能及更高效的交互性。

## xls格式文件写

```
1 1.使用xlwt模块,下载pip install xlwt
2 2.步骤:
3 workbook=xlwt.Workbook(encoding='utf8')#创建文件对象
4 worksheet=workbook.add_sheet("页面名字")
5 worksheet.write(0,0,label="hello")#0,0表示单元格的坐标,表示最左上角的单元格
6 workbook.save("文件名.xls")
7
8
9 import xlwt
10 #写
11 def funcXls():
12     #创建一个xls对象
13     wb=xlwt.Workbook(encoding='utf8')
14     #创建一个页面
15     wh=wb.add_sheet("Maker")
16     wh.write(0,0,label="hello")
17     wb.save("1.xls")
18
19 # funcXls()
```

## xls格式文件读

```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 #====#====#====#====
4 #Author:
5 #CreatDate:
6 #Version:
7 #====#====#====#====
8
9
10 #读
11 import xlrd
12 def funcRxls():
13     #打开xls文件
14     f=xlrd.open_workbook("1.xls")
15     #获取页面对象
```

```

16     sheet=f.sheets()[0]
17     for i in range(0,sheet.nrows):
18         rows=sheet.row_values(i)#获取这行内容,以列表形式返回
19         print(rows)
20
21 funcRxls()

```

## xlsx格式文件写(重点)

```

1  1.需要下载:pip install openpyxl或pyopenxl
2  注意:openpyxl只能操作xlsx格式的文件,不能操作xls格式文件

```

```

3
4  一.简单写
5  from openpyxl import workbook
6
7  def mytest01():
8      #创建工作表对象
9      wb=workbook()
10     #获取默认的面
11     mysheet=wb.active
12     mysheet['C1']=666
13     wb.save("1.xlsx")
14
15 mytest01()

```

```

16
17 二,添加页面
18 from openpyxl import workbook
19
20 def mytest01():
21     #创建工作表对象
22     wb=workbook()
23     #获取默认的面
24     mysheet=wb.active
25     mysheet['C1']=666
26     #创建页面,追加方式
27     wb.create_sheet("mysheet")
28     #创建页面并指定位置
29     wb.create_sheet("mysheet2",0)
30     wb.save("1.xlsx")
31
32 mytest01()

```

```

33
34 三.页面操作
35 from openpyxl import workbook
36
37 def mytest01():
38
39     wb=workbook()
40     wb.create_sheet('mysheet')
41     print(wb.sheetnames)
42     #通过页面名获取页面
43     # st=wb["mysheet"]

```

```

44     st=wb.get_sheet_by_name("mysheet")
45     st['B1']=1111
46     print(st.max_row)#最大的行
47     print(st.max_column)#最大的列
48
49     wb.remove(st)#删除页面
50     wb.save("1.xlsx")
51
52 mytest01()
53
54 四.单元格操作
55 from openpyxl import workbook
56
57 def mytest01():
58
59     wb=workbook()
60     wb.create_sheet('mysheet')
61     print(wb.sheetnames)
62     #通过页面名获取页面
63     # st=wb["mysheet"]
64     st=wb.get_sheet_by_name("mysheet")
65     st['B1']=1111
66     #通过行和列数添加内容到单元格
67     st.cell(1,2,"hello")
68     #写入多个单元格,追加形式,一行中多个单元格
69     st.append([4,5,6])
70     st.append([10, 20, 30])
71     #写公式
72     st['A4']="=sum(A2:A3)"
73
74
75     wb.save("1.xlsx")
76
77 mytest01()
78

```

## xlsx格式文件读(重点)

```

1  xlsx读还是openpyxl模块,但要引入这个模块的load_workbook
2
3  #!/usr/bin/env python
4  # -*- coding:utf-8 -*-
5  #====#====#====#====
6  #Author:
7  #CreatDate:
8  #Version:
9  #====#====#====#====
10 from openpyxl import load_workbook
11
12 #打开文件
13 wb=load_workbook('1.xlsx')
14 #获取表格中所有的页面名

```

```

15 mylist=wb.sheetnames
16 print(mylist)
17 #获取页表
18 wh=wb[mylist[1]]
19 #获取单元格的内容
20 print(wh['B1'].value)
21 print(wh.cell(1,2).value)
22 #遍历页面中所有的内容
23 for row in wh.rows:
24     for i in row:
25         print(i.value)

```

## Json介绍

```

1  JSON ( JavaScript Object Notation , JavaScript对象表示法 ) 是一种轻量级的数据交换语言
2
3  JSON是独立于语言的文本格式，JSON 数据格式与语言无关
4
5  JSON 数据格式的特点：
6  对象表示为键值对
7  数据由逗号分隔
8  花括号保存对象
9  对象：一个对象包含一系列非排序的名称 / 值对，一个对象以{开始，并以}结束。每个名称 / 值对之间使用:分割
10 方括号保存数组，
11 数组：一个数组是一个值的集合，一个数组以[开始，并以]结束。数组成员之间使用,分割
12

```

## JSON 数据的写入(重点)

```

1  json.dump    将dict数据写入json文件中
2  json.dumps   对数据进行编码,将python中的字典 转换为 字符串
3
4  import json
5
6  #字典
7  data={"id":"33445566",'姓名':"maker","地址":"深圳"}
8  with open("data.json",'w',encoding='utf8') as f:
9      json.dump(data,f,ensure_ascii=False,indent=4)
10
11 #把字典转换为字符串
12 mystr=json.dumps(data,ensure_ascii=False,indent=4)
13 print(mystr)
14 print(type(mystr))
15
16
17 方法中每个参数的作用
18 参数                                作用
19 skipkeys                            如果 skipkeys 是 true ( 默认为 False ) , 那么那些不是基本对象
    ( 包括 str, int, float, bool, None ) 的字典的键
    会被跳过 ; 否则引发一个 TypeError。

```

20

21 `ensure_ascii` 如果 `ensure_ascii` 是 `true` (即默认值), 输出保证将所有输入的非 ASCII 字符转义。如果 `ensure_ascii` 是 `false`, 这些字符会原样输出。

22

23 `check_circular` 如果 `check_circular` 是为假值 (默认为 `True`), 那么容器类型的循环引用检验会被跳过并且循环引用会引发一个 `OverflowError` (或者更糟的情况)。

24

25 `allow_nan` 如果 `allow_nan` 是 `false` (默认为 `True`), 那么在对严格 JSON 规格范围外的 `float` 类型值 (`nan`、`inf` 和 `-inf`) 进行序列化时会引发一个 `ValueError`。如果 `allow_nan` 是 `true`, 则使用它们的 JavaScript 等价形式 (`NaN`、`Infinity` 和 `-Infinity`)。

26

27 `indent` 如果 `indent` 是一个非负整数或者字符串, 那么 JSON 数组元素和对象成员会被美化输出为该值指定的缩进等级。如果缩进等级为零、负数或者 `""`, 则只会添加换行符。None (默认值) 选择最紧凑的表示。使用一个正整数会让每一层缩进同样数量的空格。如果 `*indent*` 是一个字符串 (比如 `"\t"`), 那个字符串会被用于缩进每一层。

28

29 `separators` 当指定时, `separators` 应当是一个 (`item_separator`, `key_separator`) 元组。当 `indent` 为 None 时, 默认值取 (`'`, `'`, `': '`), 否则取 (`'`, `'`, `': '`)。为了得到最紧凑的 JSON 表达式, 你应该指定其为 (`'`, `'`, `'`)

## JSON 数据的读取(重点)

```

1  json.load  打开json文件, 并把字符串转换为python的dict数据
2  json.loads 对数据进行解码, 将 字符串 转换为 python中的字典
3
4  import json
5  with open("data.json","r",encoding='utf8') as f:
6      d=json.load(f)
7
8  print(d)
9  print(type(d))
10
11 #注意,这里要单包双,不能双包单
12 mystr='{"ID":1,"name":"maker"}'
13 d2=json.loads(mystr)
14 print(d2)
15 print(type(d2))
16
17
18
19

```

## Json与类对象(后期讲完面向对象后讲)

```
1  一.将类实例转化成json字符串
2
3
4
5
6  二.JSON反序列化类对象
7  json反序列化是将json数据或者字符串转化为Python对象
8
9
10
11
```

## Yaml文件格式介绍(重点)

```
1  yaml是一种类似于xml以及json这种键值对类型的文件，不过它的数据展示更加直观，更容易被识别和解析出来。
   而它和python这种脚本特征的语言具有很强的交互性
2  Yaml格式文件通常用来编写项目配置，也可用于数据存储
3
4  YAML 语法
5  支持的数据类型：字典、列表、字符串、布尔值、整数、浮点数、Null、时间等
6  基本语法规则：
7
8  1、大小写敏感
9
10 2、使用缩进表示层级关系
11
12 3、相同层级的元素左侧对齐
13
14 4、键值对用冒号“:”结构表示，冒号与值之间需用空格分隔
15
16 5、数组前加有“-”符号，符号与值之间需用空格分隔
17
18 6、None值可用null 和 ~ 表示
19
20 7、多组数据之间使用3横杠---分割
21
22 8、# 表示注释，但不能在一段代码的行末尾加 #注释，否则会报错
23
24 9.不要使用tab键作为缩进，有时可能出错
25
26 python没有自带的处理yaml文件的库，需要下载第三方库PyYAML
27 pip install pyyaml
28
```

## Yaml文件写(重点)

```
1  单组数据写入使用yaml.dump()方法，加入allow_unicode=True参数防止写入的中文乱码
2  多组数据写入使用yaml.dump_all()方法
3
4  #!/usr/bin/env python
```

```

5 # -*- coding:utf-8 -*-
6 #====#====#====#====
7 #Author:
8 #CreatDate:
9 #Version:
10 #====#====#====#====
11
12 import yaml
13 mydata={
14     "id": 1,
15     "地区": "深圳",
16     "data": [{
17         "id": 1,
18         "名字": "南山区"
19     },
20     {
21         "id": 2,
22         "名字": "福田区"
23     },
24     {
25         "id": 3,
26         "名字": "光明区"
27     }]
28 }
29 with open("1.yaml",'w',encoding='utf8') as f:
30     yaml.dump(data=mydata,stream=f,allow_unicode=True)
31
32
33 多组数据:
34 #!/usr/bin/env python
35 # -*- coding:utf-8 -*-
36 #====#====#====#====
37 #Author:
38 #CreatDate:
39 #Version:
40 #====#====#====#====
41
42 import yaml
43 mydata={
44     "id": 1,
45     "地区": "深圳",
46     "data": [{
47         "id": 1,
48         "名字": "南山区"
49     },
50     {
51         "id": 2,
52         "名字": "福田区"
53     },
54     {
55         "id": 3,
56         "名字": "光明区"
57     }]

```

```

58 }
59
60 mydata2={
61     "id": 2,
62     "地区": "深圳",
63     "data": [{
64         "id": 1,
65         "名字": "南山区"
66     },
67     {
68         "id": 2,
69         "名字": "福田区"
70     },
71     {
72         "id": 3,
73         "名字": "光明区"
74     }]
75 }
76 with open("1.yaml",'w',encoding='utf8') as f:
77     yaml.dump_all(documents=[mydata,mydata2],stream=f,allow_unicode=True)
78
79
80

```

## Yaml文件读(重点)

```

1
2 读取数据使用load函数
3 读取多组数据使用，yaml.load_all()方法，返回结果为一个生成器，需要使用for循环语句获取每组数据
4
5 result = yaml.load(f.read(), Loader=yaml.FullLoader)
6 Loader=yaml.FullLoader参数不写的话对结果不会有影响，但运行时会出现警告信息。
7
8 假设有mydata.yaml文件，内容如下：
9 #正常数据
10 ip: 127.0.0.1
11 port: 3306
12 #嵌套数据
13 data:
14     user: root
15     password: 123456
16     userdb: ~
17 #列表数据
18 data2:
19     - maker
20     - 28
21     - 98.9
22 #元祖数据
23 地区: !!python/tuple
24     - 深圳

```



```
25 - 上海
26 import yaml
27
28 with open("1.yaml",'r',encoding='utf8') as f:
29     res=yaml.load(f.read(),Loader=yaml.FullLoader)
30
31 print(res)
32 print(type(res))
33
34
35 多组数据读取
36 mydatas.yaml内容如下:
37 data:
38 - ip: 127.0.0.1
39   port: 3300
40 - ip: 192.168.33.44
41   port: 2789
42 - ip: 10.25.33.24
43   port: 2020
44 id: 1
45 环境: 开发
46 ---
47 data:
48 - ip: 127.0.0.1
49   port: 3300
50 - ip: 192.168.33.45
51   port: 2789
52 - ip: 10.25.33.25
53   port: 2020
54 id: 2
55 环境: 测试
56
57 import yaml
58
59 with open("1.yaml",'r',encoding='utf8') as f:
60     res=yaml.load_all(f.read(),Loader=yaml.FullLoader)
61     print(res)
62     for i in res:
63         print(i)
64
65
66
67
```

;