Excel文件介绍

```
excel文件格式有2种，一种是xls,一种是xlsx
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.使用xlwd模块,下载pip install xlwt
2.步骤:
workbook=xlwt.Workbook(encoding='utf8')#创建文件对象
worksheet=workbook.add_sheet("页面名字")
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
workbook.save("文件名.xls")


import xlwt
#写
def funcXls():
    #创建一个xls对象
    wb=xlwt.Workbook(encoding='utf8')
    #创建一个页面
    wh=wb.add_sheet("Maker")
    wh.write(0,0,label="hello")
    wb.save("1.xls")

# funcXls()
```

xls格式文件读

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====


#读
import xlrd
def funcRxls():
    #打开xls文件
    f=xlrd.open_workbook("1.xls")
    #获取页面对象
```

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

funcRxls()
```

xlsx格式文件写(重点)

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

一.简单写
from openpyxl import Workbook

def mytest01():
    #创建工作表对象
    wb=Workbook()
    #获取默认的页面
    mysheet=wb.active
    mysheet['C1']=666
    wb.save("1.xlsx")

mytest01()

二,添加页面
from openpyxl import Workbook

def mytest01():
    #创建工作表对象
    wb=Workbook()
    #获取默认的页面
    mysheet=wb.active
    mysheet['C1']=666
    #创建页面,追加方式
    wb.create_sheet("mysheet")
    #创建页面并指定位置
    wb.create_sheet("mysheet2",0)
    wb.save("1.xlsx")

mytest01()

三.页面操作
from openpyxl import Workbook

def mytest01():

    wb=Workbook()
    wb.create_sheet('mysheet')
    print(wb.sheetnames)
    #通过页面名获取页面
    # st=wb["mysheet"]
```

```
    st=wb.get_sheet_by_name("mysheet")
    st['B1']=1111
    print(st.max_row)#最大的行
    print(st.max_column)#最大的列

    wb.remove(st)#删除页面
    wb.save("1.xlsx")

mytest01()

四.单元格操作
from openpyxl import Workbook

def mytest01():

    wb=Workbook()
    wb.create_sheet('mysheet')
    print(wb.sheetnames)
    #通过页面名获取页面
    # st=wb["mysheet"]
    st=wb.get_sheet_by_name("mysheet")
    st['B1']=1111
    #通过行和列数添加内容到单元格
    st.cell(1,2,"hello")
    #写入多个单元格,追加形式,一行中多个单元格
    st.append([4,5,6])
    st.append([10, 20, 30])
    #写公式
    st['A4']="=sum(A2:A3)"


    wb.save("1.xlsx")

mytest01()
```

xlsx格式文件读(重点)

```
xlsx读还是openpyxl模块,但要引入这个模块的load_workbook

#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
from openpyxl import load_workbook

#打开文件
wb=load_workbook('1.xlsx')
#获取表格中所有的页面名
```

```
mylist=wb.sheetnames
print(mylist)
#获取页表
wh=wb[mylist[1]]
#获取单元格的内容
print(wh['B1'].value)
print(wh.cell(1,2).value)
#遍历页面中所有的内容
for row in wh.rows:
    for i in row:
        print(i.value)
```

Json介绍

```
JSON ( JavaScript Object Notation , JavaScript对象表示法 ) 是一种轻量级的数据交换语言

JSON是独立于语言的文本格式， JSON 数据格式与语言无关

JSON 数据格式的特点:
对象表示为键值对
数据由逗号分隔
花括号保存对象
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
方括号保存数组，
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

```

JSON 数据的写入(重点)

```
json.dump    将dict数据写入json文件中
json.dumps   对数据进行编码,将python中的字典 转换为 字符串

import json

#字典
data={"id:":"33445566",'姓名:':"maker","地址:":"深圳"}
with open("data.json",'w',encoding='utf8') as f:
    json.dump(data,f,ensure_ascii=False,indent=4)

#把字典转换为字符串
mystr=json.dumps(data,ensure_ascii=False,indent=4)
print(mystr)
print(type(mystr))


方法中每个参数的作用
参数                                    作用
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。

```

```
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。

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）。

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）。

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）选择最紧凑的表达。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。

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)

```
json.load    打开json文件，并把字符串转换为python的dict数据
json.loads   对数据进行解码,将 字符串 转换为 python中的字典

import json
with open("data.json","r",encoding='utf8') as f:
    d=json.load(f)

print(d)
print(type(d))

#注意,这里要单包双,不能双包单
mystr='{"ID":1,"name":"maker"}'
d2=json.loads(mystr)
print(d2)
print(type(d2))
```

Json与类对象(后期讲完面向对象后讲)

```
一.将类实例转化成json字符串




二.JSON反序列化类对象
json反序列化是将json数据或者字符串转化为Python对象



```

Yaml文件格式介绍(重点)

```
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
Yaml格式文件通常用来编写项目配置，也可用于数据存储

YAML 语法
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
基本语法规则：

1、大小写敏感

2、使用缩进表示层级关系

3、相同层级的元素左侧对齐

4、键值对用冒号 “:” 结构表示，冒号与值之间需用空格分隔

5、数组前加有 “-” 符号，符号与值之间需用空格分隔

6、None值可用null 和 ~ 表示

7、多组数据之间使用3横杠---分割

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

9.不要使用tab键作为缩进，有时可能出错

python没有自带的处理yaml文件的库，需要下载第三方库PyYAML
pip install pyyaml

```

Yaml文件写(重点)

```
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
多组数据写入使用yaml.dump_all()方法

#!/usr/bin/env python
```

```
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====

import yaml
mydata={
    "id": 1,
    "地区": "深圳",
    "data": [{
        "id": 1,
        "名字": "南山区"
        },
        {
         "id": 2,
        "名字": "福田区"
        },
        {
        "id": 3,
        "名字": "光明区"
        }]
}
with open("1.yaml",'w',encoding='utf8') as f:
    yaml.dump(data=mydata,stream=f,allow_unicode=True)


多组数据:
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====

import yaml
mydata={
    "id": 1,
    "地区": "深圳",
    "data": [{
        "id": 1,
        "名字": "南山区"
        },
        {
         "id": 2,
        "名字": "福田区"
        },
        {
        "id": 3,
        "名字": "光明区"
        }]
```

```
}

mydata2={
    "id": 2,
    "地区": "深圳",
    "data": [{
        "id": 1,
        "名字": "南山区"
        },
        {
         "id": 2,
        "名字": "福田区"
        },
        {
        "id": 3,
        "名字": "光明区"
        }]
}
with open("1.yaml",'w',encoding='utf8') as f:
    yaml.dump_all(documents=[mydata,mydata2],stream=f,allow_unicode=True)


```

Yaml文件读(重点)

```

读取数据使用load函数
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

result = yaml.load(f.read(), Loader=yaml.FullLoader)
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。

假设有mydata.yaml文件，内容如下:
#正常数据
ip: 127.0.0.1
port: 3306
#嵌套数据
data:
  user: root
  password: 123456
  userdb: ~
#列表数据
data2:
  - maker
  - 28
  - 98.9
#元祖数据
地区: !!python/tuple
- 深圳
```

```
- 上海
import yaml

with open("1.yaml",'r',encoding='utf8') as f:
    res=yaml.load(f.read(),Loader=yaml.FullLoader)

print(res)
print(type(res))


多组数据读取
mydatas.yaml内容如下：
data:
- ip: 127.0.0.1
  port: 3300
- ip: 192.168.33.44
  port: 2789
- ip: 10.25.33.24
  port: 2020
id: 1
环境: 开发
---
data:
- ip: 127.0.0.1
  port: 3300
- ip: 192.168.33.45
  port: 2789
- ip: 10.25.33.25
  port: 2020
id: 2
环境: 测试

import yaml

with open("1.yaml",'r',encoding='utf8') as f:
    res=yaml.load_all(f.read(),Loader=yaml.FullLoader)
    print(res)
    for i in res:
        print(i)
```

;