csv文件

csv文件简介

参看 RFC 4180
http://www.ietf.org/rfc/rfc4180.txt

逗号分隔值Comma-Separated Values。
CSV 是一个被行分隔符、列分隔符划分成行和列的文本文件。
CSV 不指定字符编码。

行分隔符为\r\n，最后一行可以没有换行符
列分隔符常为逗号或者制表符。
每一行称为一条记录record

字段可以使用双引号括起来，也可以不使用。如果字段中出现了双引号、逗号、换行符必须使用双引号括起来。如果字段的值是双引号，使用两个双引号表示一个转义。

表头可选，和字段列对齐就行了。

手动生成csv文件

```
from pathlib import Path

p = Path('o:/tmp/mycsv/test.csv')
parent = p.parent
if not parent.exists():
    parent.mkdir(parents=True)

csv_body = '''\
id,name,age,comment
1,zs,18,"I'm 18"
2,ls,20,"this is a ""test"" string."
3,ww,23,"你好

计算机
"
'''
p.write_text(csv_body)
```

csv 模块

`reader(csvfile, dialect='excel', **fmtparams)`
返回reader对象，是一个**行迭代器**。

默认使用excel方言，如下：

- delimiter 列分隔符,逗号
- lineterminator 行分隔符\r\n

- quotechar 字段的引用符号，缺省为`"`双引号
- 双引号的处理
    - doublequote 双引号的处理，默认为True。如果碰到**数据中**有双引号，而quotechar也是双引号，True则使用2个双引号表示，False表示使用转义字符将作为双引号的前缀
    - escapechar 一个转义字符，默认为None
    - writer = csv.writer(f, doublequote=False, escapechar='@') 遇到双引号，则必须提供转义字符
- quoting 指定双引号的规则
    - QUOTE_ALL 所有字段
    - QUOTE_MINIMAL特殊字符字段，Excel方言使用该规则
    - QUOTE_NONNUMERIC非数字字段
    - QUOTE_NONE都不使用引号。

`writer(csvfile, dialect='excel', **fmtparams)`
返回DictWriter的实例。
主要方法有writerow、writerows。

`writerow(iterable)`

```
import csv
from pathlib import Path

p = Path('o:/tmp/mycsv/test.csv')
with open(str(p)) as f:
    reader = csv.reader(f)
    print(next(reader))
    print(next(reader))
    for line in reader:
        print(line)

rows = [
    [4,'tom',22,'tom'],
    (5,'jerry',24,'jerry'),
    (6,'justin',22,'just\t"in'),
    "abcdefghi",
    ((1,),(2,))
]
row = rows[0]

with open(str(p), 'a') as f:
    writer = csv.writer(f)
    writer.writerow(row)
    writer.writerows(rows)
```

说明row行，需要一个可迭代对象就可以，可迭代的每一个元素，将作为csv行的每一个元素。
windows下在会在每行末尾多出一个\r，解决办法`open('test.csv', 'w', newline='')`

ini文件处理

作为配置文件，ini文件格式的很流行。

```
[DEFAULT]
a = test

[mysql]
default-character-set=utf8

[mysqld]
datadir =/dbserver/data
port = 33060
character-set-server=utf8
sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES
```

中括号里面的部分称为section，译作节、区、段。
每一个section内，都是key=value形成的键值对，key称为option选项。

注意这里的DEFAULT是缺省section的名字，必须大写。

configparser

configparser模块的ConfigParser类就是用来操作。

可以将section当做key，section存储着键值对组成的字典，可以把ini配置文件当做一个嵌套的字典。默认使用的是有序字典。

`read(filenames, encoding=None)`
读取ini文件，可以是单个文件，也可以是文件列表。可以指定文件编码。

`sections()` 返回section列表。缺省section不包括在内。
`add_section(section_name)` 增加一个section。
`has_section(section_name)` 判断section是否存在

`options(section)` 返回section的所有option，会追加缺省section的option
`has_option(section, option)` 判断section是否存在这个option

`get(section, option, *, raw=False, vars=None[, fallback])`
从指定的段的选项上取值，如果找到返回，如果没有找到就去找DEFAULT段有没有。

`getint(section, option, *, raw=False, vars=None[, fallback])`
`getfloat(section, option, *, raw=False, vars=None[, fallback])`
`getboolean(section, option, *, raw=False, vars=None[, fallback])`
上面3个方法和get一样，返回指定类型数据。

`items(raw=False, vars=None)`
`items(section, raw=False, vars=None)`
没有section，则返回所有section名字及其对象；如果指定section，则返回这个指定的section的键值对组成二元组。

`set(section, option, value)`
section存在的情况下，写入option=value，要求option、value必须是字符串。

`remove_section(section)`
移除section及其所有option
`remove_option(section, option)`
移除section下的option。

`write(fileobject, space_around_delimiters=True)`
将当前config的所有内容写入fileobject中，一般open函数使用w模式。

```
from configparser import ConfigParser

filename = 'test.ini'
newfilename = 'mysql.ini'

cfg = ConfigParser()
read_ok = cfg.read(filename)
print(read_ok)
print(cfg.sections())  #
print(cfg.has_section('client'))

for k,v in cfg.items(): # 未指定section
    print(k, type(k))
    print(v, type(v))
    print(cfg.items(k))
    print()

print('-'*30)

for k,v in cfg.items("mysqld"): # 指定section
    print(k, type(k))
    print(v, type(v))
    print('~~~~~~')
print('-'*30)

# 取值
tmp = cfg.get('mysqld', 'port')
print(type(tmp), tmp)
print(cfg.get('mysqld', 'a'))
# print(cfg.get('mysqld','magedu'))
print(cfg.get('mysqld', 'magedu', fallback='python')) # 缺省值
# 按照类型
tmp = cfg.getint('mysqld', 'port')
print(type(tmp), tmp)

if cfg.has_section('test'):
    cfg.remove_section('test')

cfg.add_section('test')
cfg.set('test', 'test1', '1')
cfg.set('test', 'test2', '2')

with open(newfilename, 'w') as f:
    cfg.write(f)
```

```python
print(cfg.getint('test', 'test2'))

cfg.remove_option('test', 'test2')

# 字典操作更简单
cfg['test']['x'] = '100'  # key不存在
cfg['test2'] = {'test2': '1000'}  # section不存在

print('x' in cfg['test'])
print('x' in cfg['test2'])

# 其他内部方式
print(cfg._dict)  # 返回默认的字典类型，默认使用有序字典<class 'collections.OrderedDict'>

for k,v in cfg._sections.items():
    print(k,v)

for k,v in cfg._sections['mysqld'].items():
    print(k,v)

# 修改后需再次写入
with open(newfilename, 'w') as f:
    cfg.write(f)
```