数据的存储

```
# -*- coding: utf-8 -*-
# @Author  : quanchenliu
# @Time    : 2024/1/28
# @Function:
```

一、TXT文本文件的存储

1、一个简单的存储示例：

```
with open('movies.txt', 'w', encoding='utf-8'):
    file.write(f'名称：{name}\n')
    file.write(f'类别：{categories}\n')
    file.write(f'上映时间：{published_at}\n')
    file.write(f'评分：{score}\n')
```

2、打开方式：

- 读：
  - r：只读；
  - rb：二进制只读（常用于二进制文件）；
  - r+：读写；
  - rb+：二进制读写；
- 写：**覆盖写**
  - w：以覆盖的方式写；
  - wb：二进制覆盖写；
  - w+：覆盖式读写；
  - wb+：二进制覆盖式读写；
- 追加：**追加写**
  - a：以追加的方式写；
  - ab：二进制追加写；
  - a+：追加式读写；
  - ab+：二进制追加式读写。

二、JSON文件存储

1、对象和数组：

- 对象是 JavaScript 中用花括号 {} 包围起来的内容，是一种键值对结构；
- 数组是 JavaScript 中用中括号 [] 包围起来的内容，是一种索引结构；
- JSON 可以由以上两种形式自由组合而成，能够嵌套无限次。

2、读取 `JSON`：

一般而言，我们从文本文件中读取字符数据并将其转化为 json 对象，然后就能够运用一系列系统方法读取 json 数据。注意：JSON 的数据需要用双引号包围起来。

- loads：用于将 JSON 文本字符串转换为 JSON 对象，常用于处理**字符串**；
- load：用于从文件对象中读取 JSON 数据并将其转换为 JSON 对象，常用于处理**文件对象**；
- get：使用 loads/load 方法得到 JSON 对象之后，可以利用**索引**获取响应的内容，并使用 `get` 方法用于**获取指定键的值**（使用圆括号 `()`）。

```
import json

with open('data.json', encoding='utf-8') as file:
    str = file.read()                # 先从 json 文件中读取文本数据，并存储为字符串 str
    data = json.loads(str)           # 将文本字符串 str 转化为 json 对象
    print(data.get['name'])


# 上述示例的更简单用法：直接用 load 方法传入文件操作对象(注意，是 load 不是 loads)
data = json.load(open('data.json', encoding='utf-8'))
print(data)
```

3、输出 `JSON`：

- dumps：将 JSON 对象转换为 JSON 格式的字符串，它接收一个 Python 对象，然后返回一个表示该对象的 JSON 字符串。
- dump：将 JSON 对象转换为 JSON 格式并将其写入文件，它接收两个参数，第一个是要写入的 Python 对象，第二个是目标文件对象。
- `dumps` 用于将 JSON 对象**转换为 JSON 字符串**，而 `dump` 用于将 JSON 对象**写入到文件中**。

```
import json

data = [
    {
    'name': 'Bob',
    'gender': 'male',
    'birthday': '1992-10-18'
    },
    {
    'name': ' 王伟 ',
    'gender': ' 男 ',
```

```
    'birthday': '1992-10-18'
    }
]

# 调用 dumps 方法将 json 对象转化为字符串
# indent=2 用于保证 json 数据的缩进格式
# ensure_ascii=False 用于保证中文字符的正常输出
with open('C:/Users/DELL/Desktop/python爬虫基
础/5.FileStorageTest/data.json', 'w', encoding='utf-8') as file:
    file.write(json.dumps(data, indent=2, ensure_ascii=False))

# 同样的，dumps 方法有与之对应的 dump 方法
json.dump(data, open('data.json', 'a', encoding='utf-8'), indent=2,
ensure_ascii=False)
```

三、CSV 文件存储

CSV 文件**以纯文本形式存储表格数据**，是一个字符序列，可以由任意数目的记录组成，各条记录以某种换行符分隔开。每条记录都由若干字段组成，字段间的分隔符是其他字符或字符串，**最常见的分隔符是逗号或制表符。**所有的记录有完全相同的字段序列，相当于一个结构化的纯文本表格形式。

1、写入列表：

（1）初始化写入对象：

```
writer = csv.writer(csvfile)
```

（2）调用 writerow / writerows **方法**写入数据：

```
# 调用 writerow 方法写入数据
writer.writerow(['id', 'name', 'age'])
writer.writerow(['10001', 'Mike', 20])
writer.writerow(['10002', 'Bob', 22])
writer.writerow(['10003', 'Jordan', 21])
```

```
# 调用 writerows 方法写入数据，同时写入多行，传入参数为二维列表
writer.writerow(['id', 'name', 'age'])
writer.writerows([['10001', 'Mike', 20], ['10002', 'Bob', 22], ['10003',
'Jordan', 21]])
```

（3）delimiter 修改列与列之间的分隔符：

```
writer = csv.writer(csvfile, delimiter=' ')  # 若要指定列与列之间的分隔符，可
以传入 delimiter 参数
```

2、写入字典：

（1）先定义字段列表，用 `fieldnames` 表示：

```
fieldnames = ['id', 'name', 'age']
```

（2）将 `fieldnames` 传给 `DictWriter` **方法**以初始化字典写入对象 writer：

```
writer = csv.DictWriter(csvfile, fieldnames=filenames)
```

（3）调用 `writeheader` **方法**写入头信息：

```
writer.writeheader()
```

（4）调用 `writerow` **方法**写入数据：

```
writer.writerow({'id': '10001', 'name': 'Mike', 'age': 20})     # 调用
writerow 方法写入相应字典(自动在行末添加换行符)
writer.writerow({'id': '10002', 'name': 'Bob', 'age': 22})
writer.writerow({'id': '10003', 'name': 'Jordan', 'age': 21})
```

（5）若要**写入中文**内容，则需要指定编码格式 **'utf-8-sig'** 而不是 'utf-8'

```
with open('C:/Users/DELL/Desktop/python爬虫基
础/5.FileStorageTest/data.csv', 'a', encoding='utf-8-sig', newline='') as
csvfile:
    fieldnames = ['id', 'name', 'age']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writerow({'id': '10005', 'name': '王伟', 'age': 22})
```

3、读出：

构造 Reader 对象，通过遍历输出文件中的每一行。若包含中文信息，需指定编码格式为 **'utf-8-sig'**

```
import csv

with open('C:/Users/DELL/Desktop/python爬虫基
础/5.FileStorageTest/data.csv', 'r', encoding='utf-8-sig') as csvfile:
    reader = csv.reader(csvfile)              # reader 是一个 Reader 对象
    for row in reader:                        # 遍历输出每一行，每一行都是一个
列表
        print(row)
```

4、借助 pandas 库完成写入/读出：

```
import pandas as pd

data = [
    {'id': '10001', 'name': 'Mike', 'age': 20},
    {'id': '10002', 'name': 'Bob', 'age': 22},
    {'id': '10003', 'name': 'Jordan', 'age': 21},
]


# 将 data 写入 CSV 文件
df = pd.DataFrame(data)                    # 新建一个 DataFrame 对象
df.to_csv('C:/Users/DELL/Desktop/python爬虫基
础/5.FileStorageTest/data.csv', index=False)

# 从 CSV 文件中读取 data
df = pd.read_csv('C:/Users/DELL/Desktop/python爬虫基
础/5.FileStorageTest/data.csv')
data = df.values.tolist()                  # 调用 .values.tolist 方法，将数据转
化为列表
print(data)
for index, row in df.iterrows():           # 对 df 进行逐行遍历，同样能够得到列
表类型的结果
    print(row.tolist())
```