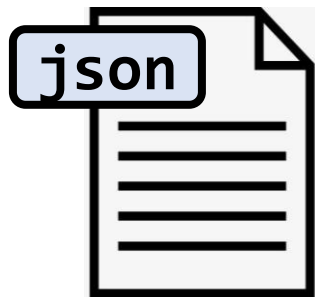




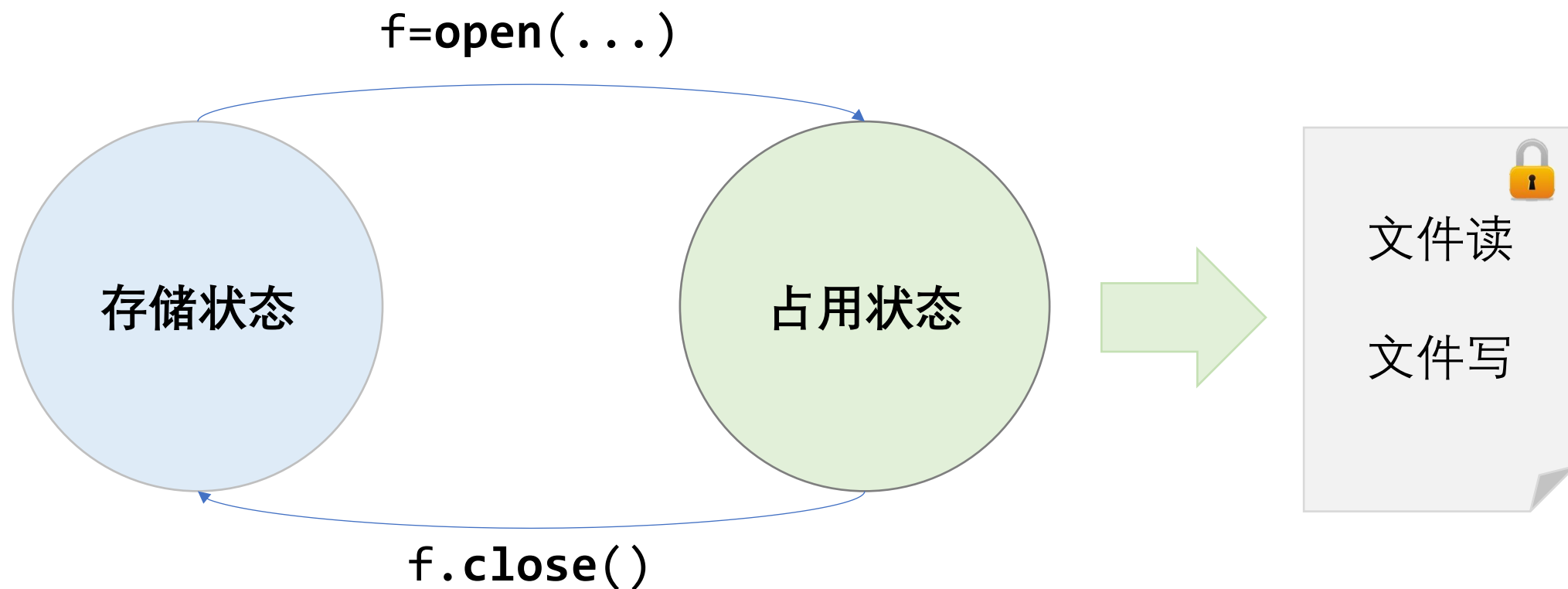
python™

程序设计

11. 文件操作



文件的状态



文件

保存在硬盘上的一组数据序列，可包含任何数据内容。

- **文本文件**：由单一特定编码(如, UTF-8)的字符组成。字符串、易于阅读，每行以'\n'结尾
- **二进制文件**：由比特0、1组成，无统一编码。以字节串bytes存储。如 jpg、mp4格式

名称	类型
 a.txt	文本文档



```
# 以文本文件方式、只读打开文件
f = open("a.txt", "rt", encoding='utf-8')
print(f.read())
f.close()
```

Python程序设计

```
# 以二进制文件方式、只读打开文件
f = open("a.txt", "rb")
print(f.read())
f.close()
```

b'Python\xe7\xa8\x8b\xe5\xba\x8f\xe8\xae\xbe\xe8\xae\xa1'

打开文件 open()

变量名 = **open**(文件名[, 打开方式])

打开方式	含 义
'r'	只读模式， 文件不存在返回异常（默认）
'w'	覆盖写模式， 文件不存在则创建， 存在则覆盖原文件
'x'	创建写模式， 文件不存在则创建， 存在则返回异常
'a'	追加写模式， 文件不存在则创建， 存在则追加
'b'	二进制文件模式
't'	文本文件模式（默认）
'+'	与 r/w/x/a一同使用， 在原功能基础上增加同时读写功能

```
# 以文本文件方式、只读打开文件，不能写入
f = open("a.txt", "r", encoding='utf-8')
print(f.read())
f.close() # 若不关闭，可能在以后有不可预期的损害
```

```
# 以文本文件方式、只读打开文件，增加写入功能
f = open("a.txt", "r+", encoding='utf-8')
f.write('2024') # 2024从文件头覆盖写入
f.close()
```

文件读

方法	含义
<code>f.read(size=-1)</code>	读入全文，返回字符串。也可以指定读入长度size
<code>f.readline(size=-1)</code>	读入一行
<code>f.readlines(hint=-1)</code>	读入所有行。也可以指定读入行数hint
<code>f.seek(offset, whence=0)</code>	定位文件位置， <code>whence=0</code> : 文件开头, <code>1</code> : 当前位置, <code>2</code> : 文件结尾
<code>f.tell()</code>	获取当前文件的读写位置

```
f = open("b.txt", "r", encoding='utf-8')
```

```
s = f.read()          # 全读入，返回字符串
```

```
print(s)              复旦大学校训：  
                      博学而笃志，切问而近思
```

```
f.seek(0)             # 返回文件头
```

```
ls = f.readlines()    # 全读入，返回列表
```

```
print(ls)             ['复旦大学校训：\n', '博学而笃志，切问而近思']
```

```
f.close()
```

```
# 用for循环一行一行读，每行结束有'\n'
```

```
f = open("b.txt", "r", encoding='utf-8')
```

```
for line in f:  
    print(line)
```

```
f.close()
```

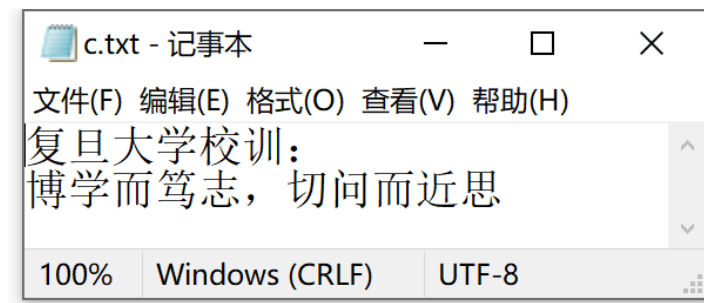
```
复旦大学校训：
```

```
博学而笃志，切问而近思
```

文件写

方法	含义
<code>f.write(s)</code>	向文件写入一个字符串/字节串
<code>f.writelines(lines)</code>	将 列表 写入文件

```
f = open("c.txt", "w", encoding='utf-8')
f.write('复旦大学校训: \n') # 结尾有换行符
f.write('博学而笃志, 切问而近思')
f.close()
```



```
ls = ['复旦大学校训: \n', '博学而笃志, 切问而近思']
f = open("c.txt", "w", encoding='utf-8')
f.writelines(ls)
f.close()
```

print()

```
with open("Hi.txt", 'w') as f:
```

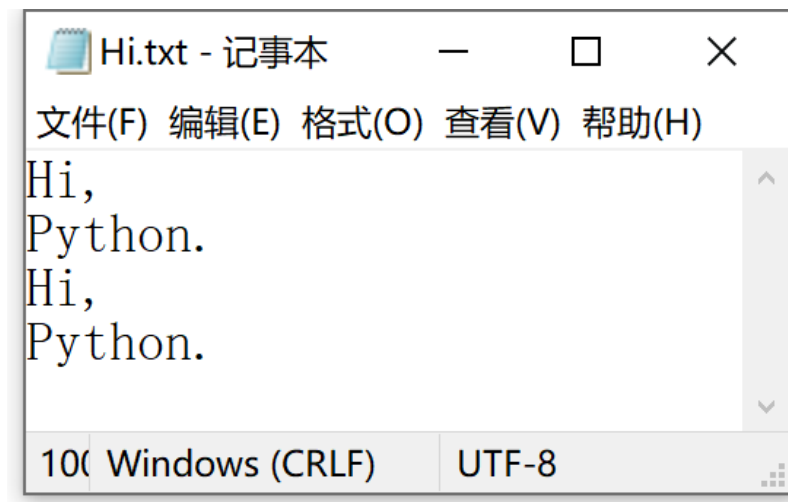
```
    f.write("Hi,\n")           # 写入第1行
```

```
    f.write("Python.\n")      # 写入第2行
```

也可以用print()

```
print("Hi,", file=f)         # 写入第3行
```

```
print("Python.", file=f)    # 写入第4行
```



关键字: `with`

不必在程序中关闭文件, **结束with时自动关闭**

```
with open(文件名[, 打开方式]) [as 文件对象]:  
    相关指令  
    ...
```

```
with open("a.txt", "r", encoding='utf-8') as f:  
    data = f.read()  
    print(data)
```

复制文件

```
with open("a.txt", "r", encoding='utf-8') as f1, open("b.txt", 'w') as f2:  
    f2.write(f1.read())
```


文件压缩库：zipFile

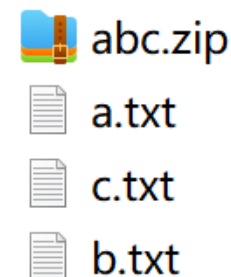
创建、读取、添加、删除以及解压缩等操作

- 压缩文件

```
import zipfile

# 需要压缩的文件列表
to_zip = ['a.txt', 'b.txt', 'c.txt']

# 创建zip文件
with zipfile.ZipFile('abc.zip', 'w') as myzip:
    for file in to_zip:
        myzip.write(file) # 添加文件压缩
```



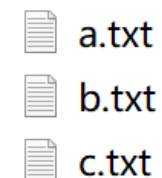
- 解压文件

```
import zipfile

# 解压zip文件
with zipfile.ZipFile('abc.zip', 'r') as myzip:
    myzip.extractall('new_dir') # 解压文件到子目录中
```

› new_dir

名称



提供方便使用操作系统相关功能的函数。

- 删除文件

```
import os

f = 'a.txt'

# 先判断是否存在，再删除文件
if os.path.exists(f):
    os.remove(f)
    print(f, '已删除。')
else:
    print(f, '不存在!')
```

- 重命名文件

```
import os

src = 'd:\\a.txt'
dst = 'd:\\1.txt'

if os.path.exists(src):
    os.rename(src, dst)
    print(src, '已重命名。')
else:
    print(src, '不存在!')
```

当前工作目录、拼接路径

```
import os
```

```
# 获取当前工作目录
```

```
path = os.getcwd()
```

```
# 文件名
```

```
file = 'a.txt'
```

```
# 拼接成一个新的路径
```

```
full = os.path.join(path, file)
```

```
print(full)
```

C:\Users\Sam2023\Desktop\a.txt

- **join()**不检测路径是否真实存在。
- 路径拼接时，不要直接用字符串拼接。
- 根据操作系统使用正确的路径分隔符 (Windows上是反斜杠 \, Unix上是正斜杠 /) 来连接路径中的各个部分，确保在不同操作系统上都能正常工作。

切分文件名

```
import os
```

```
path = "d:/Python课件/01简介.pptx"
```

```
# os.path.split(): 切分为路径名、文件全名
```

```
dirName, fileName = os.path.split(path)
```

```
print('路径名: ', dirName)
```

路径名: d:/Python课件

```
print('文件全名: ', fileName)
```

文件全名: 01简介.pptx

```
# os.path.splitext(): 切分为文件名、扩展名
```

```
file_name, file_extension = os.path.splitext(fileName)
```

```
print('文件名: ', file_name)
```


文件名: 01简介

```
print('扩展名: ', file_extension)
```

扩展名: .pptx

(D:) > Python课件

名称

 01简介.pptx

目录操作

- 创建目录

```
import os

path = 'c:\\demo'

# 先判断是否存在, 再创建
if not os.path.exists(path):
    os.mkdir(path)
    print(path, '已创建。')
else:
    print(path, '已存在!')
```

- 删除目录

```
import os

path = 'c:\\demo'

# 先判断是否存在, 再删除
if os.path.exists(path):
    os.rmdir(path)
    print(path, '已删除。')
else:
    print(path, '不存在!')
```

遍历目录下的文件

`os.listdir()` : 返回指定的文件夹包含的文件或文件夹的名字的列表。

```
import os
```

```
# 打开文件夹
```

```
path = "d:/Python课件/"
```

```
dirs = os.listdir( path )
```

```
# 输出所有文件和文件夹
```

```
for file in dirs:
```

```
    print(file)
```

(D:) > Python课件

名称

01 简介.pptx

02 变量、运算符、基本数据类型、内置函数.pptx

03 选择结构、内置模块.pptx

04 循环结构.pptx

05 字符串.pptx

06 列表、元组.pptx

07 绘图库turtle.pptx

08 函数1.pptx

09 函数2.pptx

10 字典、集合.pptx

01 简介.pptx

02 变量、运算符、基本数据类型、内置函数.pptx

03 选择结构、内置模块.pptx

04 循环结构.pptx

05 字符串.pptx

06 列表、元组.pptx

07 绘图库turtle.pptx

08 函数1.pptx

09 函数2.pptx

10 字典、集合.pptx

shutil 模块

提供文件、文件夹的复制、移动等操作。

- 复制文件

```
import shutil
```

```
src = r'c:\a.txt'
```

```
dst = r'd:' # 仅复制
```

```
dst = r'd:\b.txt' # 复制+修改文件名
```

```
shutil.copy(src, dst)
```

- 移动文件

```
import shutil
```

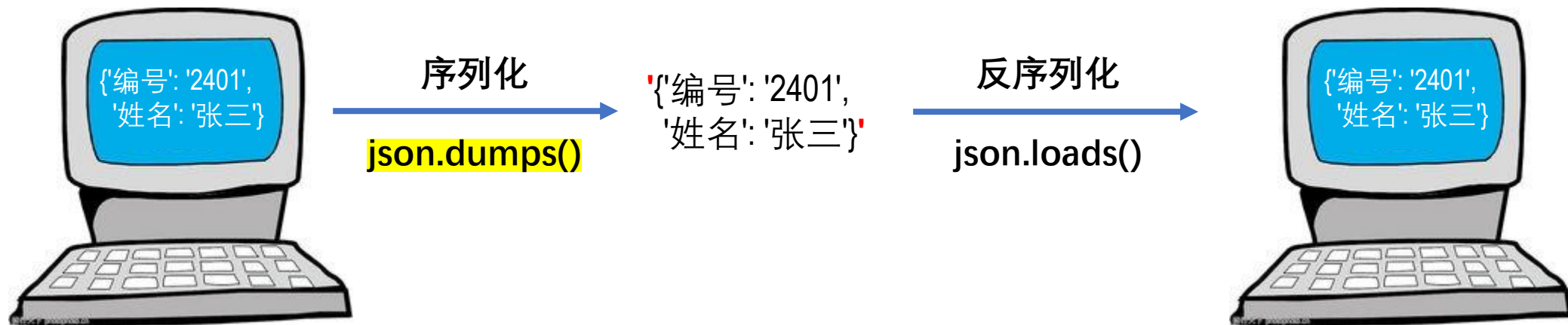
```
src = r'd:\a.txt'
```

```
dst = r'd:\temp' # 目标文件夹要存在
```

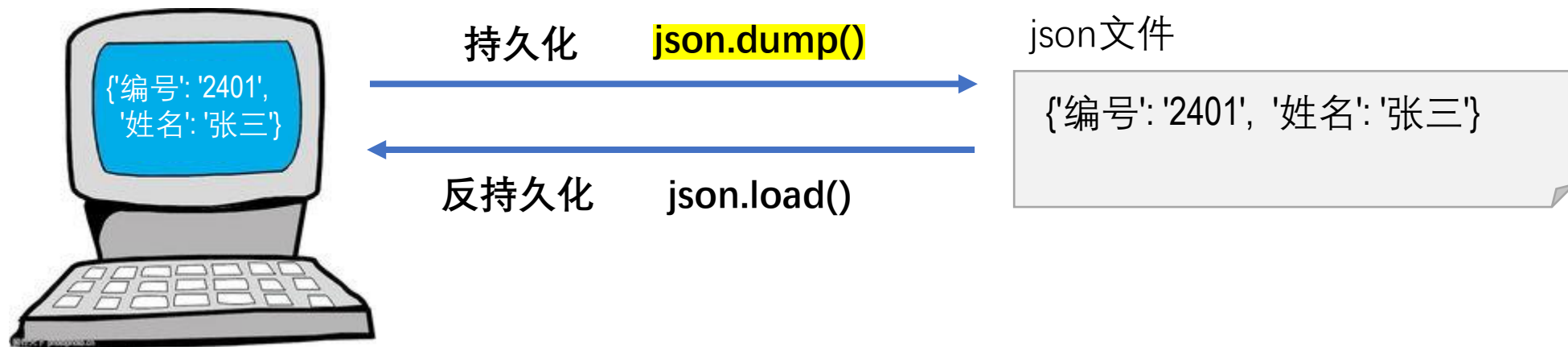
```
dst = r'd:\temp\b.txt' # 更改目标文件名
```

```
shutil.move(src, dst) # 移动后删除原文件
```

- **序列化**：将内存中的对象转换为字符串/字节串。用于数据交换，如，计算机间的网络编程。



- **持久化**：将内存中的对象转换为字符串/字节串后，再写入文件。如，退出游戏前保存游戏的状态。

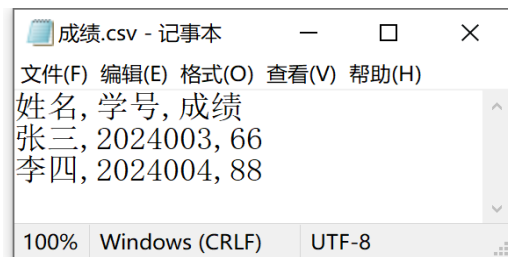


CSV (逗号分隔值) 格式

Comma-Separated Values

- 以纯文本形式存储表格数据的简单文件格式。
- 只支持简单的表格结构，每一行代表一个数据记录，每一列代表一个数据字段，一般用逗号分隔。
- 优点：简单。值没有类型，全是字符串。没有多个工作表，不支持多表格、复杂数据类型。

• 写入CSV文件



```
import csv
```

```
f = '成绩.csv'
```

```
with open(f, 'w', encoding='utf-8', newline='') as cf:
```

```
    w = csv.writer(cf)
```

```
    w.writerow(['姓名', '学号', '成绩'])
```

```
    w.writerow(['张三', '2024003', '66'])
```

```
    w.writerow(['李四', '2024004', '88'])
```

• 读取CSV文件

```
import csv
```

```
f = '成绩.csv'
```

```
with open(f, encoding='utf-8') as cf:
```

```
    r = csv.reader(cf)
```

```
    for row in r:
```

```
        print(row) # 每行是一个列表
```

```
['姓名', '学号', '成绩']
```

```
['张三', '2024003', '66']
```

```
['李四', '2024004', '88']
```

二维数据的读取

	A	B	C
1	姓名	学号	成绩
2	张三	2024003	66
3	李四	2024004	88

```
import csv
```

```
f = '成绩.csv'
```

```
with open(f, encoding='utf-8') as cf:
```

```
    r = csv.DictReader(cf)    # 将每行中的信息映射到一个字典
```

```
    for row in r:
```

```
        print(row)    # 每行是一个字典
```

```
{'姓名': '张三', '学号': '2024003', '成绩': '66'}
```

```
{'姓名': '李四', '学号': '2024004', '成绩': '88'}
```

- 一种轻量级的数据交换格式。常用于接口数据传输。
- 易于人类阅读和编写，也易于机器解析和生成。

- 序列化

```
import json
```

```
d = {'编号': '2401', '姓名': '张三'}
j = json.dumps(d, ensure_ascii=False) # 转存
print(j) # 一般用True, 不必显示出来看
{"编号": "2401", "姓名": "张三"}
```

```
j = json.dumps(d, ensure_ascii=False, indent=4)
print(j) # 带有缩进, 易于阅读
{
    "编号": "2401",
    "姓名": "张三"
}
```

Python 数据类型	json 数据类型
True	true
False	false
None	null
int, float, long	number
str, unicode	string
list, tuple	array
dict	object

- 反序列化

```
import json
```

```
d = json.loads(j) # s: string
print(d)
```

```
{'编号': '2401', '姓名': '张三'}
```

json 持久化

- 写入 json 文件

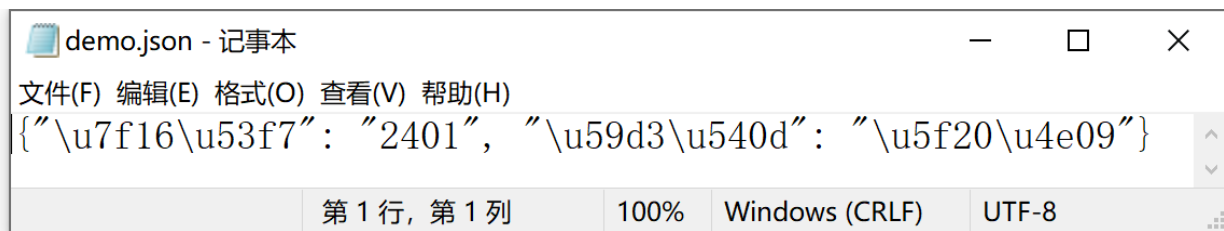
```
import json
```

```
d = {'编号': '2401', '姓名': '张三'}
```

```
f = 'demo.json'
```

```
with open(f, 'w') as fj:
```

```
    json.dump(d, fj)
```



- 读出 json 文件

```
import json
```

```
f = 'demo.json'
```

```
with open(f, 'r') as fj:
```

```
    d = json.load(fj)
```

```
print(d)
```

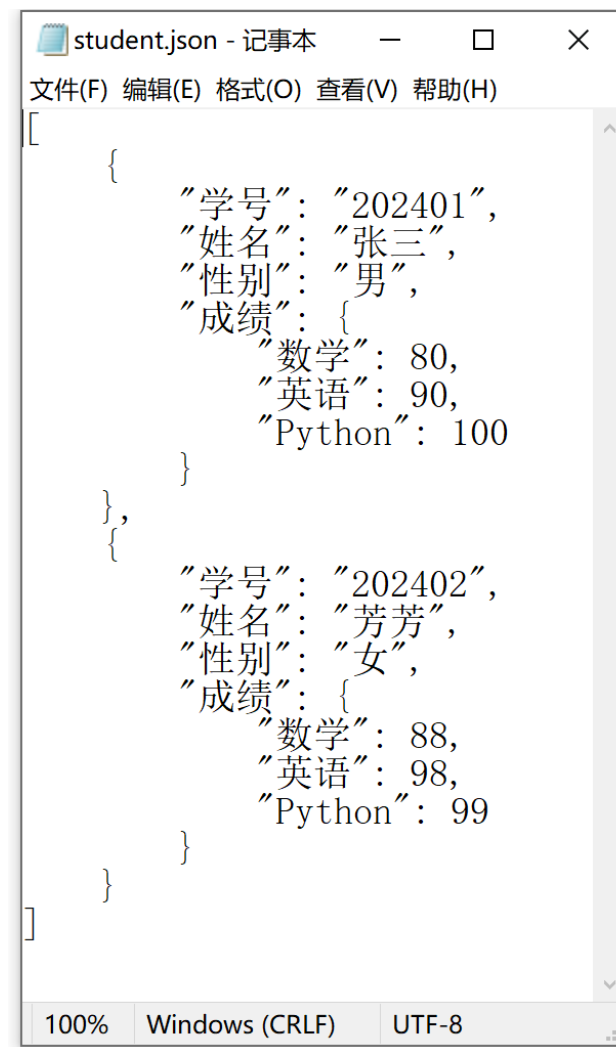

```
{'编号': '2401', '姓名': '张三'}
```

例：将变量保存到json文件中

```
import json

student = [
    {
        '学号': '202401',
        '姓名': '张三',
        '性别': '男',
        '成绩': {
            '数学': 80,
            '英语': 90,
            'Python': 100
        }
    },
    {
        '学号': '202402',
        '姓名': '芳芳',
        '性别': '女',
        '成绩': {
            '数学': 88,
            '英语': 98,
            'Python': 99
        }
    }
]

f = 'student.json'
with open(f, 'w', encoding='utf-8') as fj:
    json.dump(student, fj, ensure_ascii=False, indent=4)
```



```
[
    {
        "学号": "202401",
        "姓名": "张三",
        "性别": "男",
        "成绩": {
            "数学": 80,
            "英语": 90,
            "Python": 100
        }
    },
    {
        "学号": "202402",
        "姓名": "芳芳",
        "性别": "女",
        "成绩": {
            "数学": 88,
            "英语": 98,
            "Python": 99
        }
    }
]
```

yaml 格式

yaml、json都是数据交换格式, **yaml用于数据结构复杂的数据交换、编写配置文件**

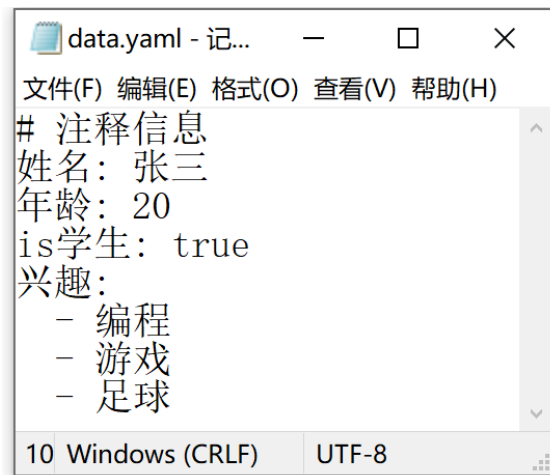
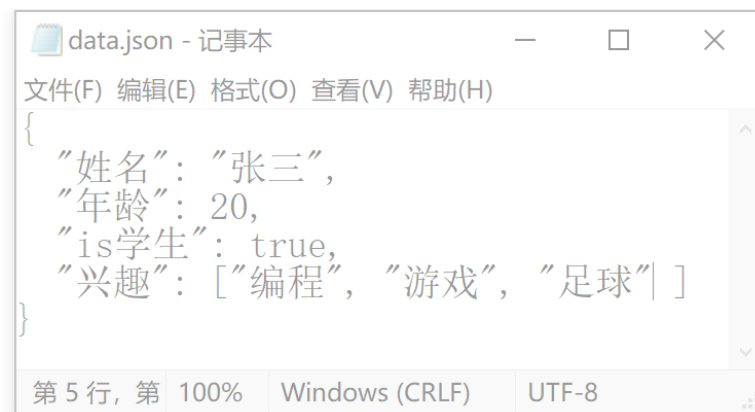
```
> pip install pyyaml
```

```
import json
with open('data.json', 'r', encoding='utf-8') as file:
    data_json = json.load(file)
    print(data_json)
```

```
{'姓名': '张三', '年龄': 20, 'is学生': True, '兴趣': ['编程', '游戏', '足球']}
```

```
import yaml
with open('data.yaml', 'r', encoding='utf-8') as file:
    data_yaml = yaml.safe_load(file) # full_load()
    print(data_yaml)
```

```
{'姓名': '张三', '年龄': 20, 'is学生': True, '兴趣': ['编程', '游戏', '足球']}
```



注释
: 字典
- 列表

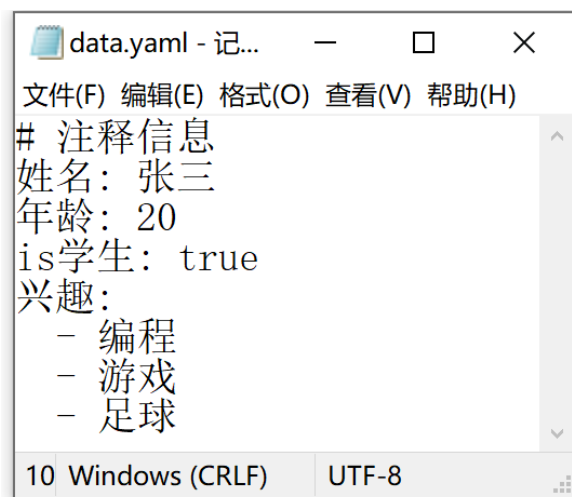
写入 yaml 文件

```
import yaml
```

```
a = {'姓名': '张三', '年龄': 20, 'is学生': True,  
     '兴趣': ['编程', '游戏', '足球']}
```

```
with open('data.yaml', 'w') as file:
```

```
    yaml.dump(a, file, encoding='utf-8', allow_unicode=True)
```



假消息生产器 库: [faker](#)

生成虚假数据, 如, 姓名、电子邮件或详细的虚假个人资料。

```
> pip install Faker
```

```
from faker import Faker
```

```
fake = Faker('zh_CN')
```

```
print(fake.name())
```

```
# 赵欢
```

```
print(fake.address())
```

```
# 河南省桂英县房山永安路x座 808964
```

```
print(fake.email())
```

```
# li68@example.org
```

```
print(fake.date())
```

```
# 2009-09-20
```

```
print(fake.url())
```

```
# https://vt.cn/
```

```
print(fake.phone_number())
```

```
# 15093295701
```

```
print(fake.user_name())
```

```
# ocheng
```

```
print(fake.company())
```

```
# 艾提科信网络有限公司
```

```
print(fake.job())
```

```
# 会务/会展经理
```