序列化和反序列化

通过文件操作，我们可以将字符串写入到一个本地文件。但是，如果是一个对象(例如列表、字典、元组等)，就无法直接写入到一个文件里，需要对这个对象进行序列化，然后才能写入到文件里。

设计一套协议，按照某种规则，把内存中的数据转换为字节序列，保存到文件，这就是序列化，反之，从文件的字节序列恢复到内存中，就是反序列化。

Python中提供了JSON和pickle两个模块用来实现数据的序列化和反序列化。

JSON模块

JSON(JavaScriptObjectNotation, JS对象简谱)是一种轻量级的数据交换格式，它基于 ECMAScript 的一个子集，采用完全独立于编程语言的**文本格式**来存储和表示数据。JSON的本质是字符串！

使用JSON实现序列化

JSON提供了dump和dumps方法，将一个对象进行序列化。

dumps方法的作用是把对象转换成为字符串，它本身不具备将数据写入到文件的功能。

```
import json
file = open('names.txt', 'w')
names = ['zhangsan', 'lisi', 'wangwu', 'jerry', 'henry', 'merry', 'chris']
# file.write(names)  出错，不能直接将列表写入到文件里

# 可以调用 json的dumps方法，传入一个对象参数
result = json.dumps(names)

# dumps 方法得到的结果是一个字符串
print(type(result))  # <class 'str'>

# 可以将字符串写入到文件里
file.write(result)

file.close()
```

dump方法可以在将对象转换成为字符串的同时，指定一个文件对象，把转换后的字符串写入到这个文件里。

```
import json

file = open('names.txt', 'w')
names = ['zhangsan', 'lisi', 'wangwu', 'jerry', 'henry', 'merry', 'chris']

# dump方法可以接收一个文件参数，在将对象转换成为字符串的同时写入到文件里
json.dump(names, file)
file.close()
```

注意：如果是一个空对象，调用dumps方法转换成为一个JSON对象，得到的结果是null(JS里的空对象)

```
json.dumps(None)  # null
```

使用JSON实现反序列化

使用loads和load方法，可以将一个JSON字符串反序列化成为一个Python对象。

loads方法需要一个字符串参数，用来将一个字符串加载成为Python对象。

```
import json

# 调用loads方法，传入一个字符串，可以将这个字符串加载成为Python对象
result = json.loads('["zhangsan", "lisi", "wangwu", "jerry", "henry", "merry", "ch
ris"]')
print(type(result))  # <class 'list'>
```

load方法可以传入一个文件对象，用来将一个文件对象里的数据加载成为Python对象。

```
import json

# 以可读方式打开一个文件
file = open('names.txt', 'r')

# 调用load方法，将文件里的内容加载成为一个Python对象
result = json.load(file)

print(result)
file.close()
```

pickle模块

和json模块类似，pickle模块也有dump和dumps方法可以对数据进行序列化，同时也有load和loads方法进行反序列化。区别在于，json模块是将对象转换成为字符串，而pickle模块是将对象转换成为二进制。

pickle模块里方法的使用和json里方法的使用大致相同，需要注意的是，**pickle是将对象转换成为二进制，所**

以，如果想要把内容写入到文件里，这个文件必须要以二进制的形式打开。

区别(了解)

> 思考: json和pickle两个模块都可以将对象进行序列化和反序列化，那它们有哪些区别，在使用场景上又该如何选择?

- json模块:
  - 将对象转换成为字符串，不管是在哪种操作系统，哪种编程语言里，字符串都是可识别的。
  - json就是用来在不同平台间传递数据的。
  - 并不是所有的对象都可以直接转换成为一个字符串，下标列出了Python对象与json字符串的对应关系。

Python	JSON
dict	object
list,tuple	array
str	string
int,float	number
True	true
False	false
None	null

- 如果是一个自定义对象，默认无法装换成为json字符串，需要手动指定JSONEncoder.
- 如果是将一个json串重新转换成为对象，这个对象里的方法就无法使用了。

```
import json
class MyEncode(json.JSONEncoder):
    def default(self, o):
        # return {"name":o.name,"age":o.age}
        return o.__dict__

class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def eat(self):
          print(self.name+'正在吃东西')

p1 = Person('zhangsan', 18)

# 自定义对象想要转换成为json字符串，需要给这个自定义对象指定JSONEncoder
result = json.dumps(p1, cls=MyEncode)
print(result)  # {"name": "zhangsan", "age": 18}

# 调用loads方法将对象加载成为一个对象以后，得到的结果是一个字典
p = json.loads(result)
print(type(p))
```

- pickle模块：
    - pickle序列化是将对象按照一定的规则转换成为二进制保存，它不能跨平台传递数据。
    - pickle的序列化会将对象的所有数据都保存。