I/O编程

本章的IO编程都是同步模式，异步IO由于复杂度太高，后续涉及到服务器端程序开发时再讨论.

1. 文件读写

读写文件是最常见的IO操作。Python内置了读写文件的函数，用法和C是兼容的.

1.1 读文件

要以读文件的模式打开一个文件对象，使用Python内置的open()函数，传入文件名和标示符：

```
f = open('test.txt', 'r')

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'test.txt'
```

```
f.read() # 一次读取文件的全部内容

f.close()
```

读写文件时可能会遇到IOError，后面的f.close()函数可能得不到执行，不能正常关闭文件。

```
try:
    f = open('/path/to/file', 'r')
    print(f.read())
finally:
    if f:
        f.close()
```

更方便地：

```
with open('/path/to/file', 'r') as f:
    print(f.read())
```

每次读一行

```
for line in f.readlines():
    print(line.strip()) # 把末尾的'\n'删掉
```

1.2 二进制文件

```
f = open('test.txt', 'rb')
f.read()
```

1.3 字符编码

要读取非UTF-8编码的文本文件，需要给open()函数传入encoding参数，例如，读取GBK编码的文件

```
f = open('test.txt', 'r', encoding='gbk')
f.read()
```

1.4 写文件

写文件和读文件是一样的，唯一区别是调用open()函数时，传入标识符'w'或者'wb'表示写文本文件或写二进制文件：

```
with open('test.txt', 'w') as f:
    f.write('Hello, world!')
```

- 要写入特定编码的文本文件，请给open()函数传入encoding参数，将字符串自动转换成指定编码。
- 可以传入'a'以追加（append）模式写入

2. StringIO和BytesIO

很多时候，数据读写不一定是文件，也可以在内存中读写。

2.1 StringIO

StringIO顾名思义就是在内存中读写str。

- 写入

要把str写入StringIO，我们需要先创建一个StringIO，然后，像文件一样写入即可：

```
>>> from io import StringIO
>>> f = StringIO()
>>> f.write('hello')
5
>>> f.write(' ')
1
>>> f.write('world!')
6
>>> print(f.getvalue())
hello world!
```

- 读取

```
>>> from io import StringIO
>>> f = StringIO('Hello!\nHi!\nGoodbye!')
>>> while True:
...     s = f.readline()
...     if s == '':
...         break
...     print(s.strip())
...
Hello!
Hi!
Goodbye!
```

2.2 BytesIO

StringIO操作的只能是str，如果要操作二进制数据，就需要使用BytesIO。

BytesIO实现了在内存中读写bytes，我们创建一个BytesIO，然后写入一些bytes：

```
>>> from io import BytesIO
>>> f = BytesIO()
>>> f.write('中文'.encode('utf-8'))
6
>>> print(f.getvalue())
b'\xe4\xb8\xad\xe6\x96\x87'
```

```
>>> from io import BytesIO
>>> f = BytesIO(b'\xe4\xb8\xad\xe6\x96\x87')
>>> f.read()
b'\xe4\xb8\xad\xe6\x96\x87'
```

3. 文件和目录

os模块的使用，请自学

4. 序列化

4.1 pickle模块

```
d = dict(name='Bob', age=20, score=88)
# {'name': 'Bob', 'age': 20, 'score': 88}
```

变量d存储在内存中。把变量从内存中变成可存储或传输的过程称之为序列化，序列化之后，就可以把序列化后的内容写入磁盘，或者通过网络传输到别的机器上。

反过来，把变量内容从序列化的对象重新读到内存里称之为反序列化.

Python提供了pickle模块来实现序列化。

```
>>> import pickle
>>> d = dict(name='Bob', age=20, score=88)
>>> pickle.dumps(d) #pickle.dumps()方法把任意对象序列化成一个bytes
b'\x80\x03}q\x00(X\x03\x00\x00\x00ageq\x01K\x14X\x05\x00\x00\x00scoreq\x02KXX\
x04\x00\x00\x00nameq\x03X\x03\x00\x00\x00Bobq\x04u.'
```

- 序列化，写入文件

```
>>> f = open('dump.txt', 'wb')
>>> pickle.dump(d, f)
>>> f.close()
```

- 反序列化

```
>>> f = open('dump.txt', 'rb')
>>> d = pickle.load(f)
```

```
>>> f.close()
```

4.2 JSON

如果要在不同的编程语言之间传递对象，就必须把对象序列化为标准格式，比如XML，但更好的方法是序列化为JSON，因为JSON表示出来就是一个字符串，可以被所有语言读取，也可以方便地存储到磁盘或者通过网络传输。JSON不仅是标准格式，并且比XML更快，而且可以直接在Web页面中读取，非常方便。

JSON表示的对象就是标准的JavaScript语言的对象.

- 序列化

Python内置的json模块提供了非常完善的Python对象到JSON格式的转换:

```
>>> import json
>>> d = dict(name='Bob', age=20, score=88)
>>> json.dumps(d)
'{"age": 20, "score": 88, "name": "Bob"}'
```

dumps()方法返回一个str，内容就是标准的JSON。类似的，dump()方法可以直接把JSON写入一个file-like Object.

- 反序列化

```
>>> json_str = '{"age": 20, "score": 88, "name": "Bob"}'
>>> json.loads(json_str)
{'age': 20, 'score': 88, 'name': 'Bob'}
```

用loads()或者对应的load()方法，前者把JSON的字符串反序列化，后者从file-like Object中读取字符串并反序列化.

- 进阶 Python的dict对象可以直接序列化为JSON的{}, 怎么把class序列化呢

```
import json

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

s = Student('Bob', 20, 88)
print(json.dumps(s))    # TypeError
```

错误是因为默认情况下，dumps()方法不知道如何将Student实例变为一个JSON的{}对象。

```
def student2dict(std):
    return {
        'name': std.name,
        'age': std.age,
        'score': std.score
    }
print(json.dumps(s, default=student2dict))

# print(json.dumps(s, default=lambda obj: obj.__dict__))
```

同样，反序列

```
def dict2student(d):
    return Student(d['name'], d['age'], d['score'])
```

```
>>> json_str = '{"age": 20, "score": 88, "name": "Bob"}'
>>> print(json.loads(json_str, object_hook=dict2student))
```

传入的object_hook函数负责把dict转换为Student实例