文件

文件类型

按照文件中数据的组织形式，文件分为文本文件和二进制文件

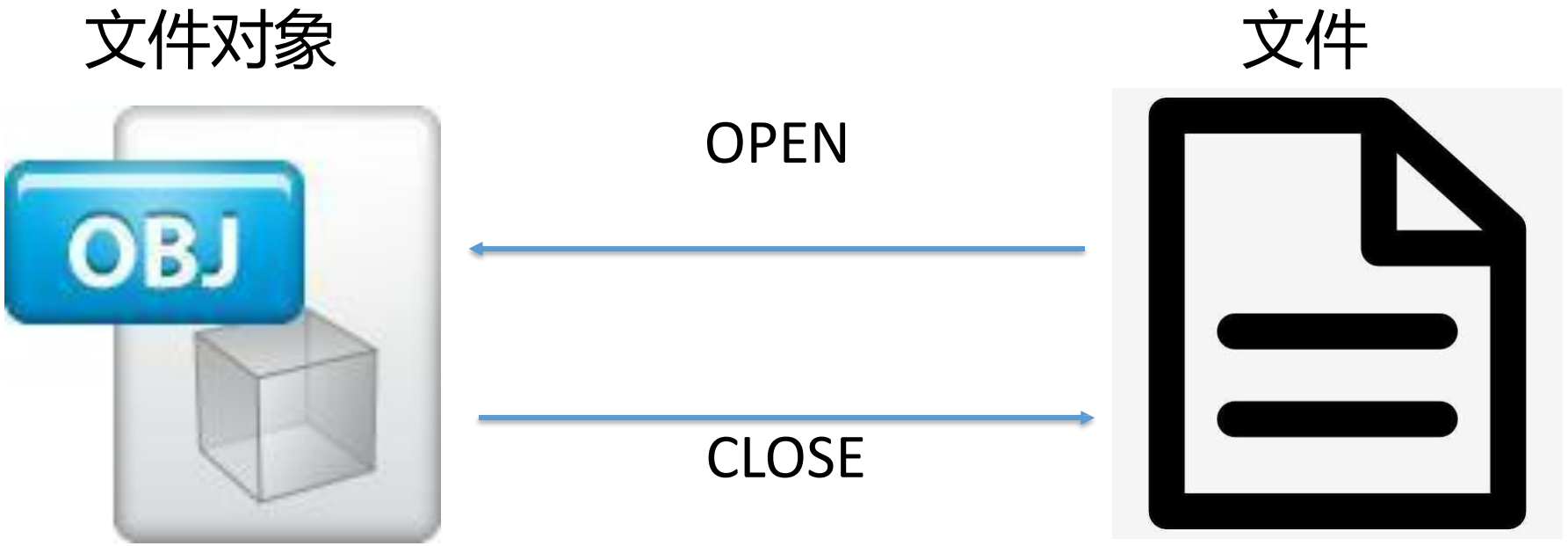
- 文本文件：存储常规字符串，由若干文本行组成。常规字符串指文本编辑器能正常显示、编辑且人能够直接阅读和理解的字符串

- 二进制文件：存储字节串（bytes）。无法用普通文本处理软件进行编辑，通常无法被人直接阅读和理解。需要使用专门的软件进行解码后读取、显示、修改或执行

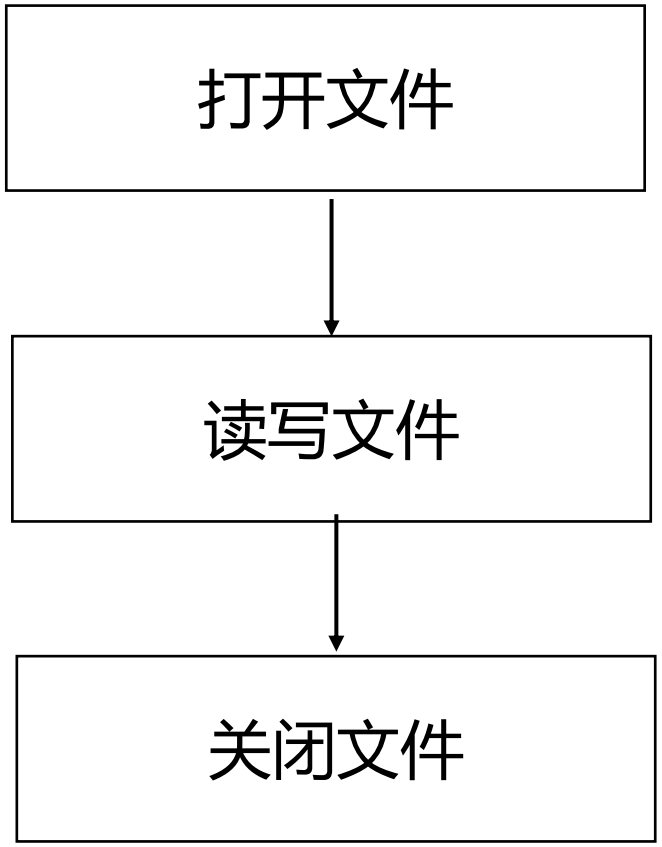
内容

- 文件的基本操作
- 二进制文件的序列化操作

文件操作的实质



文件的基本操作流程



#打开文件，创建文件对象

#读取、写入、删除、修改

#关闭文件，保存文件内容

打开指定文件

```
fp=open(file, mode='r', buffering=-1,
        encoding=None, errors=None,
        newline=None, closfd=True,
        opener=None)
```

文件打开模式

模式	说明
r	只读方式打开文件，若文件不存在则报错，默认模式
w	只写方式打开文件，若文件存在则覆盖
a	追加方式打开文件，在文件尾部追加写入
x	独占方式创建文件，若文件存在则报错
t	文本模式（需与前面模式组合使用），默认模式
b	二进制模式（需与前面模式组合使用）
+	读写模式（需与前面模式组合使用）

文本文件读出操作

以‘r’、‘r+’、‘w+’、‘a+’方式打开的文本文件，可执行读出操作

方法	说明
fp.read([size])	读出size个字符的内容作为结果返回，不指定size表示读出全部内容
fp.readline([size])	读出一行内容作为结果返回，指定size时与read()方法相同
fp.readlines()	把每行文本作为一个字符串存入列表中，返回该列表
fp.readable()	测试当前文件是否可读，返回True或False

文本文件读出操作

Tiny grass,
your steps are small,
but you possess the earth
under your tread.

tinyGrass.txt

```
>>> f1 = open('tinyGrass.txt', 'r')
>>> s1 = f1.read(6)
>>> print(s1, end = '***')
Tiny g***
>>> s2 = f1.readline()
>>> print(s2, end = '***')
rass,
***
>>> s3 = f1.readlines()
>>> for s in s3:
...     print(s, end = '***')
your steps are small,
***but you possess the earth
***under your tread.
***
```

文本文件写入操作

以'w'、'a'、'x'、'w+'、'a+'、'x+'、'r+'方式打开的文本文件，可执行写入操作

方法	说明
fp.write (s)	把s的内容写入文件
fp.writelines(s)	把字符串列表写入文本文件，不添加换行符
fp.flush()	强制立即将缓冲区的内容写入文件
fp.writable()	测试当前文件是否可写，返回True或False

文本文件写入操作

```
>>> s1 = 'Twinkle, twinkle, little star'
>>> s2 = 'How I wonder what you are'
>>> s3 = ['Up above the world so high', 'Like a diamond in the sky']
>>> f3 = open('littleStar.txt', 'w')
>>> f3.write(s1)
29
>>> f3.write(s2)
25
>>> f3.writelines(s3)
>>> f3.close()
```

文本文件写入操作

```
>>> f4 = open('littleStar.txt')
>>> s4 = f4.readlines()
>>> for s in s4:
...     print(s)
...
...

Twinkle, twinkle, little starHow I wonder what you areUp above the world
so highLike a diamond in the sky
```

文件内容的随机访问

方法	说明
fp.seek (offset[, whence])	把文件指针移动到新的位置，offset表示相对于whence的位移（字节数）。whence的值：0——从文件头开始；1——从当前位置开始；2——从文件尾开始。默认为0
fp.tell()	返回文件指针的当前位置
fp.truncate([size])	删除从当前指针位置到文件末尾的内容。如果指定了size，则不论指针在什么地方，只留下前size个字节，其余内容一律删除
fp.seekable()	测试当前文件是否支持随机访问，返回True或False

文件内容的随机访问

随机访问函数可以同时应用于以文本模式或二进制模式打开的文件，但在应用于文本模式打开的文件时，会受到限制

Tiny grass,
your steps are small,
but you possess the earth
under your tread.

```
>>> f5 = open('tinyGrass.txt')
>>> f5.read(6)
'Tiny g'
>>> f5.tell()
6
```

```
>>> f5.seek(8, 0)
8
>>> f5.readline()
'ss,\n'
>>> f5.tell()
13
```

文件内容的随机访问

随机访问函数可以同时应用于以文本模式或二进制模式打开的文件，但在应用于文本模式打开的文件时，会受到限制

Tiny grass,
your steps are small,
but you possess the earth
under your tread.

```
>>> f5.seek(1, 1)
Traceback (most recent call last):
  File "<pyshell#107>", line 1, in <module>
    f5.seek(1, 1)
io.UnsupportedOperation: can't do nonzero cur-relative seeks
>>> f5.seek(0, 1)
13
>>> f5.seek(0, 2)
83
```

文件的关闭

fp.close()

在进行文件操作时，如果文件读写后，未能正确关闭一个文件，有可能导致文件损坏、文件内容丢失等问题

出现异常时无法正常关闭文件

```
f4 = open("test.txt","r+")
exit(-1)  #模拟程序异常退出
f4.close() #此句永远无法执行到
```

程序执行过程中异常退出时，无法执行后面的close语句

文件操作的简化方式：with语句

```
with open(file, mode= ‘r’ , buffering=-1,
          encoding=None, errors=None,
          newline=None, closfd=True,
          opener=None) as fp:
```

<文件操作>

with关键字用于对资源进行访问的场合，会确保不管在使用过程中是否发生异常，都会在使用结束后执行必要的清理操作，释放资源，自动关闭文件等

文件对象常用属性

属性	说明
buffer	文件的缓冲区对象
closed	文件是否关闭
fileno	文件的文件号，一般不需要
mode	文件的打开模式
name	文件名称

内容

- 文件的基本操作
- 二进制文件的序列化操作

Python对象的序列化（Serialization）

- 序列化：将内存对象的状态信息转换为可以存储或传输的形式的过程
- 反序列化：将文件中存储或网络传输的对象转换为内存对象的过程

- 常用的Python序列化模块（方法）：pickle, json，marshal, shelve, struct

作用：内存对象的持久保存和跨越机器/平台的传输

pickle模块

- pickle是Python专用的序列化模块，不能跨平台
- pickle可实现对Python所有数据类型（对象）结构的二进制序列化和反序列化过程

pickle基本方法

方法	说明
pickle.dump (obj, file[, …])	将obj封存后的对象写入已打开的文件file
pickle.dumps(obj[, …])	将obj封存后的对象作为字节类型直接返回，而不是将其写入文件
pickle.load(file[…])	从已打开的文件file中读取封存后的对象，重建其中特定对象的层次结构并返回
pickle.loads(data[…])	从data中读取封存后的对象，重建其中的对象层次结构并返回。data必须是类字节对象

pickle封存操作

```
import pickle

year = 2022
pi = 3.1415926
s = "天地玄黄，宇宙洪荒"
lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
tu = (3, 5, 7, 11)
col = {-1, 0, 1}
dict1= {'a':'apple', 'b':'banana', 'g':'grape'}
data = [year, pi, s, lst, tu, col, dict1]

with open('pickle_sample.dat', 'wb') as fp:
    try:
        for item in data:
            pickle.dump(item, fp)
    except:
        print('写文件异常！')
```

Pickle解封操作

```
import pickle

with open('pickle_sample.dat','rb') as fp:
    while True:
        try:
            item = pickle.load(fp)
            print(item)
        except EOFError:
            break
```

```
2022
3.1415926
天地玄黃，宇宙洪荒
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
(3, 5, 7, 11)
{0, 1, -1}
{'a': 'apple', 'b': 'banana', 'g': 'grape'}
```

文件：总结

- 文件操作在各类软件开发中都占有重要地位
- 文件操作通过文件对象的打开、读写及关闭进行
- 文件的读写方法都会自动改变文件对象中指针的位置
- 对于一般的二进制文件，无法直接读取和理解其内容，往往需要通过专门的第三方库对其进行操作
- 通过序列化和反序列化操作，可实现内存对象（包括其数据和结构）的持久化或跨平台传输