



Chap7 Files

---

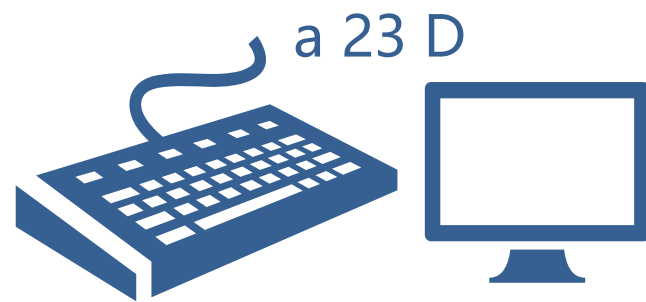
# 第7章 文件

---

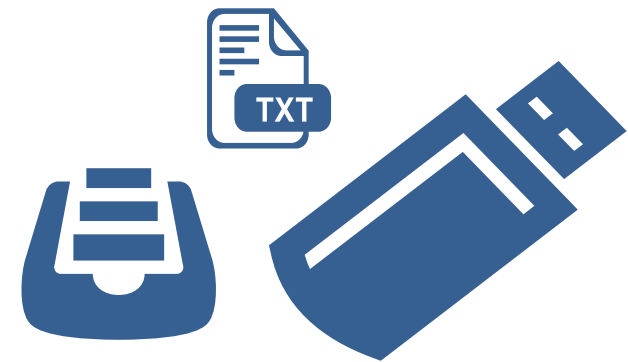
Department of Computer Science and Technology  
Department of University Basic Computer Teaching  
Nanjing University

# 程序中的数据

2



or



# 7.1

## 文件基本概念

# 文件基本概念

- 文件：存储在某种介质上的信息集合
- 存储：外部存储介质（硬盘）
- 识别：文件名
- 分类
  - 存取方式：顺序存取，随机存取
  - 文件内容表示方式：二进制文件，文本文件



## 7.1.1 Python文件系统

## 二进制文件与文本文件

12345的内存存储形式 

00110000	00111001
----------	----------

↓ 转换成ASCII编码形式

00110001	00110010	00110011	00110100	00110101
----------	----------	----------	----------	----------

↓ 以ASCII编码形式写入fp

00110001	00110010	00110011	00110100	00110101
----------	----------	----------	----------	----------

  
fp 对应的文件

-----  
12345的内存存储形式 

00110000	00111001
----------	----------

↓ 不进行转换直接写入fp

00110000	00111001
----------	----------

  
fp 对应的文件

## 二进制文件与文本文件

- 用二进制形式输出时

- 可节省外存空间和转换时间
- 一个字节并不对应一个字符，不能直接输出字符形式。
- 可读性差，常用于保存中间结果数据和运行程序。

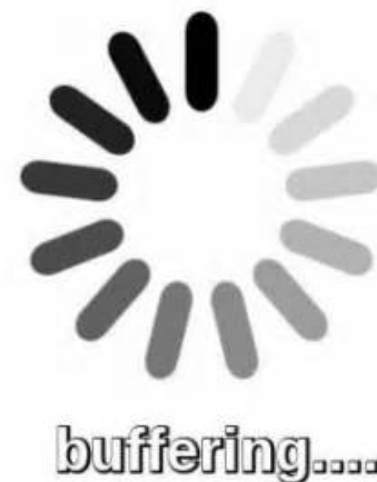


- 文本形式输出时

- 一个字节与一个字符一一对应
- 便于对字符进行逐个处理，也便于输出字符；
- 占存储空间较多；
- 要花费转换时间。

## 二进制文件与文本文件

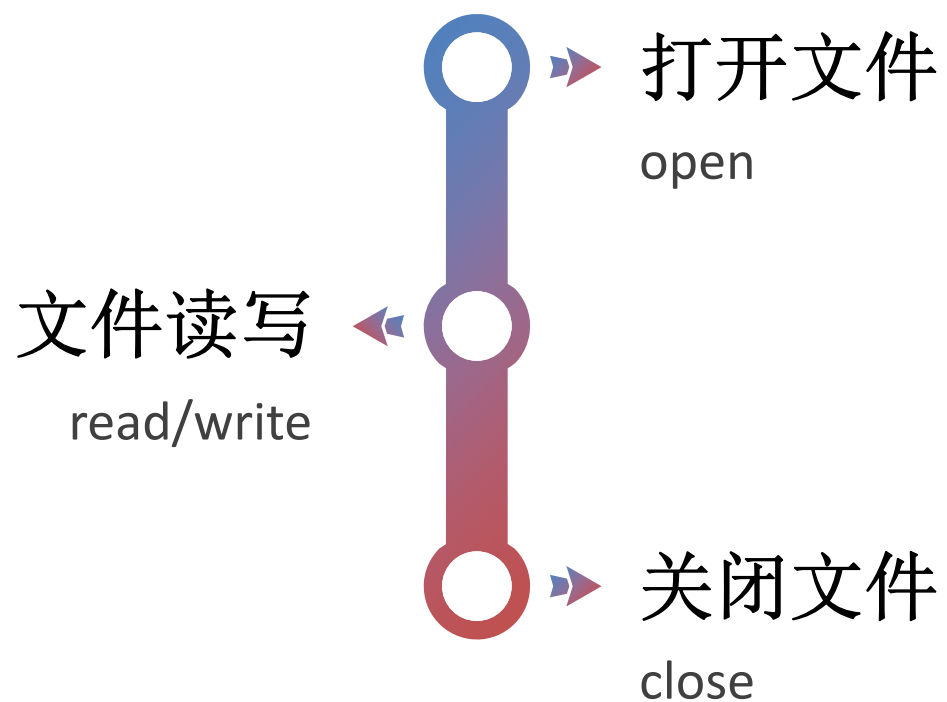
- Python中可以处理二进制文件以及文本文件
- 对二进制文件的操作可以选择是否使用缓冲区
- 文本文件均使用缓冲区处理
- 缓冲区是内存中的区域，由于内存与内存之间的数据交换比内存与外存储器之间的数据交换速度快，因此当程序中需要进行频繁的文件读写操作时，使用缓冲区可以减少I/O操作从而提高效率，也方便管理





## 7.1.2 文件的使用过程

# 文件的使用过程



## 7.2

# 文件的打开和关闭

## 7.2.1 文件的打开

```
fp = open(file, mode='r', buffering=-1)
```

- 文件名的路径：
  - DOS和Windows下，路径分隔符用 “\\” 或用 “r” 指明为原始字符串，如 “d:\\f.txt” 或 “r'd:\\f.txt”
  - Linux和Mac系统中，路径用 “/” 分隔，如 “/User/DZ/data”
  - “/” 也可以在Windows中作为路径分隔符用

# 文件的打开

```
fp = open(file, mode='r', buffering=-1)
```

Source

```
>>> root_dir = r'd:\pyprogs'  
>>> fp = open(root_dir + r'\a.txt')
```

绝对路径

Source

```
>>> f1 = open('a.txt')  
>>> f2 = open('pics/tfile.txt')  
>>> f3 = open('../a.txt')
```

相对路径

# open()函数-mode参数

15

Mode	Function
r	以读模式打开，文件必须存在
w	以写模式打开，若文件不存在则新建文件，否则清空原内容
x	以写模式打开，若文件已经存在则失败
a	以追加模式打开，若文件存在则向结尾追加内容，否则新建文件
b	以二进制模式打开，可添加到其他模式中使用
t	以文本模式打开（默认），可添加到其他模式中使用
+	以读/写模式打开，可添加到其他模式中使用

## 文件的打开

```
fp = open(file, mode='r', buffering=-1)
```

- mode为可选参数，默认值为 "r"
- buffering也为可选参数，默认值为-1，用于设置缓冲策略的可选整数
- 其他常用参数：encoding（指定编码字符集）



# 文件的打开

- open()函数返回一个文件 (file) 对象
- 默认状态是以'r'与't'模式打开



```
>>> fp = open('a.txt')  
>>> fp = open('record.data' , 'wb')
```

# 文件打开模式

18

Mode	Function
r+	以读写模式打开，必须打开已经存在的文件，读写从头部开始
w+	以读写模式打开，新建或是清空原内容，读的内容为新写入内容
a+	以读和追加模式打开，若打开已经存在的文件，读写从尾部开始
wb	以二进制写模式打开（参见w）
rb+	以二进制读写模式打开（参见r+），可写成rb+、r+b、+rb
wb+	以二进制读写模式打开（参见w+）

- 当程序启动后，以下三种标准文件有效


标准文件	文件对象	对应的设备
标准输入	stdin	键盘
标准输出	stdout	显示器
标准错误	stderr	显示器

## 7.2.2 文件的关闭

# 关闭文件

`fp.close()`

- fp为文件对象
- 关闭文件是打开文件的逆操作，切断文件对象与外存储器中文件之间的联系

 `>>> fp = open(r'd:\nfile.txt', 'r')`  
`>>> type(fp)`  
`<class '_io.TextIOWrapper'>`  
`>>> fp.name`  
`'d:\nfile.txt'`  
`>>> fp.mode`  
`'r'`  
`>>> fp.closed`  
`False`  
`>>> fp.close()`  
`>>> fp.closed`  
`True`

- 文件使用完后如果不关闭，则当程序运行结束时由系统自动关闭
- 不建议使用系统自动关闭的原因
  - 操作系统允许程序同时打开的文件个数是**有限**的
  - 写入内容已处理完若缓冲区还未满，缓冲区的内容要等到程序运行结束时由系统自动关闭该文件后才能写出，此时若系统发生非正常情况当前缓冲区中的未写到外存储上的内容就**可能丢失**掉

## 7.3

# 文件的基本操作

# 文件的基本操作

- 根据打开文件模式对文件进行 读/写 操作
- 常见文件操作函数
  - `read()`, `readline()`, `readlines()`, `write()`, `writelines()`
  - `seek()`



## 7.3.1 文件的读写

- `open()`函数返回一个文件 (file) 对象
- 文件对象可迭代, 是一个迭代器

```
for line in fp:  
    对line进行处理
```

## 读文件-read()方法

```
s = fp.read(size)
```

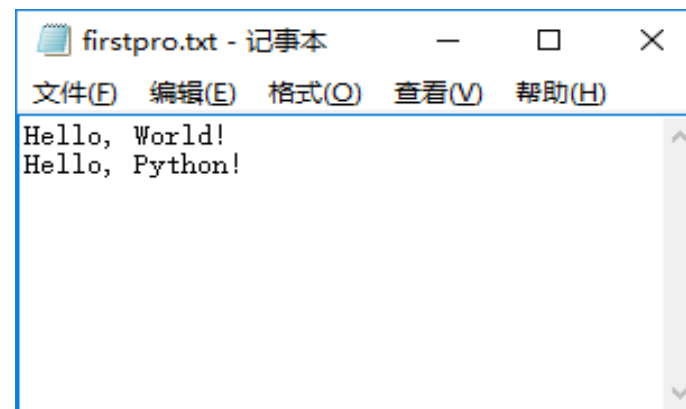
- fp为读模式打开的文件对象，文本文件或二进制文件均可
- size为从文件当前位置读取的字节数，若size为负数或空，则读取到文件结束（读到文件末尾时返回空字符串）
- 返回一个字符串（文本文件）或字节流（二进制文件）

# 读文件-read()方法

28



```
>>> fp = open('firstpro.txt')
>>> s = fp.read(5)
>>> print(s)
Hello
>>> s = fp.read()
>>> s
', World!\nHello, Python!'
>>> fp.close()
```



## 读文件-readline()方法

```
s = fp.readline(size=-1)
```

- size为从文件当前位置读取本行内的字节数，若size为默认值或大小超过当前位置到行尾字符长度，则读取到本行结束（包含换行符）
- 返回读取到的字符串内容

```
firstpro.txt :  
Hello, World!  
Hello, Python!
```

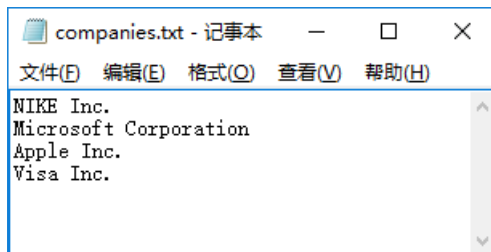


```
>>> fp = open('firstpro.txt')  
>>> s = fp.readline(20)  
>>> s  
'Hello, World!\n'  
>>> s = fp.readline(2)  
>>> s  
'He'  
>>> s = fp.readline()  
>>> s  
'llo, Python!'  
>>> fp.close()
```

# 读文件-readlines()方法

```
lines = fp.readlines(hint=-1)
```

- hint为从文件当前读写位置开始读取需要的字节数，至少为一行；若hint为默认值或负数，则读取从当前位置到文件末尾的所有行（包含换行符）
- 返回从文件中读出的行组成的列表



```
>>> fp = open('companies.txt')
>>> lines = fp.readlines(2)
>>> lines
['NIKE Inc.\n']
>>> lines = fp.readlines()
>>> lines
['Microsoft Corporation\n', 'Apple Inc.\n', 'Visa Inc.']
>>> fp.close()
```

## 写文件-write()方法

`fp.write(s)`

- 向文件中写入数据（字符串或字节流）
- 返回写入的字符数或字节数



```
>>> fp = open('firstpro.txt', 'w')
```

```
>>> fp.write("Hello, World!\n")
```

```
14
```

```
>>> fp.write("Hello, Python!")
```

```
14
```

```
>>> fp.close()
```

## 写文件-write()方法

`fp.write(s)`

- 向文件中写入数据（字符串或字节流）
- 返回写入的字符数或字节数



```
>>> f = open('firstpro.dat', 'wb')
>>> x = bytes([3, 4, 5])
>>> f.write(x)
>>> f.close()
```



## 写文件-writelines()方法

`fp.writelines(lines)`

- 向文件中写入列表数据，多用于文本文件

 Source

```
>>> fp = open('companies1.txt', 'w')
>>> lines = ['NIKE Inc.\n', 'Microsoft Corporation\n', 'Apple Inc.\n', 'Visa Inc.\n']
>>> fp.writelines(lines)
>>> fp.close()
```

## 例7.1 文件读写例子



将文件companies.txt 的字符串前加上序号1、2、3、...后写到另一个文件scompanies.txt中。

**F**<sub>ile</sub>

```
# prog7-1.py
f1= open('companies.txt')
lines = f1.readlines()
f1.close()
for i in range(len(lines)):
    lines[i] = str(i+1) + ' ' + lines[i]
f2 = open('scompanies.txt', 'w')
f2.writelines(lines)
f2.close()
```

### Output:

```
1 NIKE Inc.
2 Microsoft Corporation
3 Apple Inc.
4 VISA Inc.
```

## 7.3.2 文件的定位

## 文件的定位-`seek()`方法

```
fp.seek(offset, whence=0)
```

- `fp`打开的文件必须允许随机访问
- 在文件中移动文件指针，从`whence`（0表示文件头部，1表示当前位置，2表示文件尾部）偏移`offset`个字节
- `whence`参数可选，默认值为0
- 返回当前的读写位置

## 文件的定位-`seek()`方法

Source

```
>>> fp = open('testseek.dat', 'wb+')
>>> fp.write(b'Hello,word!')
11
>>> fp.seek(0)
0
>>> s = fp.read(5)
>>> s
b'Hello'
>>> fp.seek(-5, 2)
6
>>> s = fp.read()
```

Source

```
>>> s
b'word!'
>>> fp.seek(3, 0)
3
>>> fp.read(3)
b'lo,'
>>> fp.seek(2, 1)
8
>>> fp.read(3)
b'rd!'
>>> fp.close()
```

## 例7.2 文件读写例子改写



将文件companies.txt 的字符串前加上序号1、2、3、...。

**F**<sub>ile</sub>

```
# prog7-2.py
f= open('companies.txt', 'r+')
lines = f.readlines()
for i in range(0, len(lines)):
    lines[i] = str(i+1) + ' ' + lines[i]
f.seek(0)
f.writelines(lines)
f.close()
```

### Output:

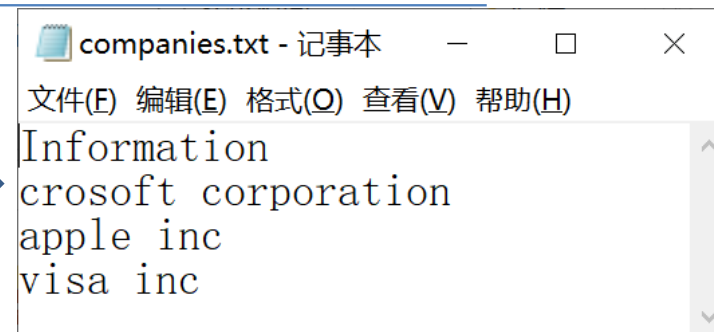
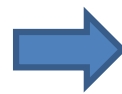
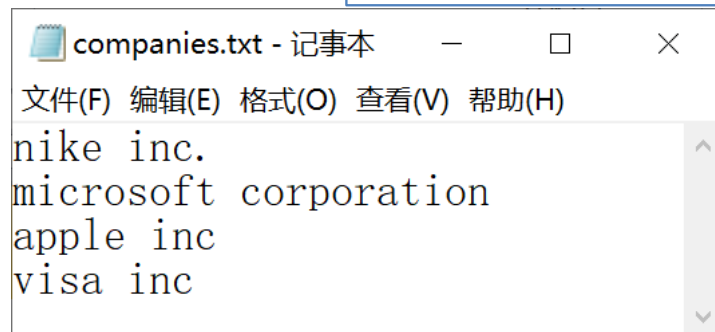
```
1 NIKE Inc.
2 Microsoft Corporation
3 Apple Inc.
4 VISA Inc.
```

## 文件中头部插入一个新行



```
f= open('companies.txt', 'r+')  
lines = f.readlines()  
f.seek(0)  
f.write('Information\n')  
f.close()
```

覆盖模式



## 文件中头部插入一个新行

File

```
f= open('companies.txt', 'r+')
lines = f.readlines()
for i in range(0, len(lines)):
    lines[i] = str(i+1) + ' ' + lines[i]
lines.insert(0, 'Information\n')
f.seek(0)
f.writelines(lines)
f.close()
```



## 文件的定位-tell()方法

### fp.tell()

- 返回文件的当前读写位置



```
>>> fp = open('testseek.dat', 'rb+')
>>> fp.tell()
0
>>> fp.read(5)
b'Hello'
>>> fp.tell()
5
>>> fp.close()
```

## 7.3.3 文件的其他操作

## 文件的其他方法及属性

方法	功能
<b>f.flush()</b>	将写缓冲区的数据写入文件
<b>f.truncate(size=None)</b>	将文件截取为给定大小的字节（如果未指定大小，则为当前文件读写位置）。当前文件读写位置没有改变
<b>f.closed</b>	文件关闭属性，当文件关闭时为True，否则为False
<b>f.fileno()</b>	返回文件描述符（整数）
<b>f.readable()</b>	判断文件是否可读，是返回True，否返回False
<b>f.writable()</b>	判断文件是否可写，是返回True，否返回False
<b>f.seekable()</b>	判断文件是否支持随机访问，是返回True，否返回False
<b>f.isatty()</b>	判断文件是否交互（如连接到一个终端设备），是返回True，否返回False

## with语句——文件异常处理



```
with open('companies.txt', 'r+') as f:
    lines = f.readlines()
    for i in range(len(lines)):
        lines[i] = str(i+1) + ' ' + lines[i]
    f.seek(0)
    f.writelines(lines)
```


## 例7.3 文件综合例



请用随机函数产生500行1-100之间的随机整数存入文件random.txt中，编程寻找这些整数的众数并输出，众数即为一组数中出现最多的数。

利用列表将对应数字出现的次数记录下来，假设数字45出现了5次，则列表lst[45]的值即为5，结果也有可能不止一个。本例也可以使用字典解决。

## 文件综合例

 `# prog7_3.py`  
`import random`  
`with open('random.txt', 'w+') as fp:`  
    `for i in range(500):`  
        `fp.write(str(random.randint(1, 100)))`  
        `fp.write('\n')`  
    `fp.seek(0)`  
    `nums = fp.readlines()`  
    `nums = [num.strip() for num in nums]`  
    `setNums = set(nums)`  
    `lst = [0] * 101`  
    `for num in setNums:`  
        `c = nums.count(num)`  
        `lst[int(num)] = c`  
    `for i in range(len(lst)):`  
        `if lst[i] == max(lst):`  
            `print(i)`

- 文件的打开
- 文件打开模式与文件类型
- 文件的关闭
- 文件的读写
- 文件定位

