

6.1 本地文件获取

文件基本概念

- 文件：存储在某种介质上的信息集合
- 存储：外部介质
- 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



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。

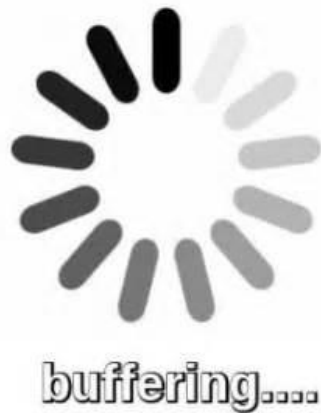


• 文本形式输出时

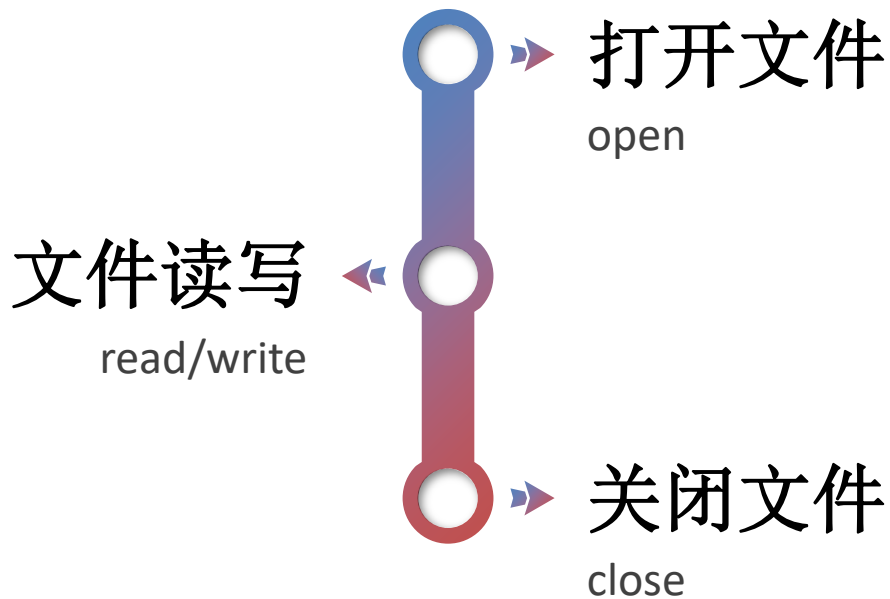
- 一个字节与一个字符——对应
- 便于对字符进行逐个处理，也便于输出字符；
- 占存储空间较多；
- 要花费转换时间。

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
- 缓冲区是内存中的区域，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
- 文本文件均使用缓冲区处理



文件的使用过程



文件的打开

Source

```
>>> f1 = open('d:\\infile.txt')
>>> f2 = open('d:/outfile.txt', 'w')
>>> f3 = open('frecord.csv', 'ab', 0)
```

open()函数返回
一个文件 (file)
对象

file_obj = open(filename, mode='r', buffering=-1)

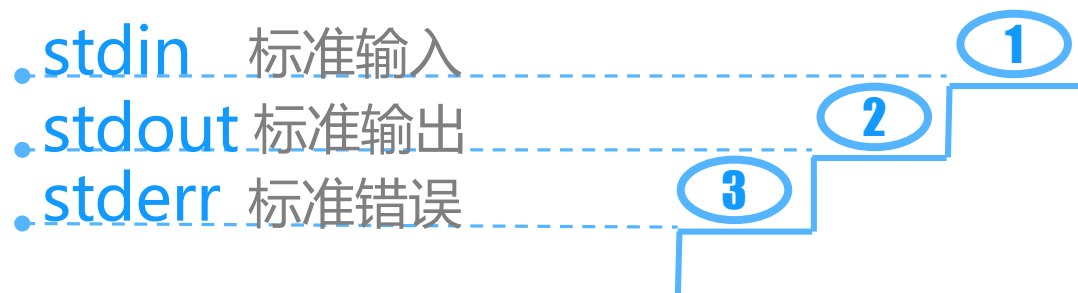
- mode为可选参数，默认值为r
- buffering也为可选参数，默认值为-1（0代表不缓冲，1或大于1的值表示缓冲一行或指定缓冲区大小）
- 其他常用参数：encoding

open()函数-mode

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
r+	以读写模式打开
w+	以读写模式打开（清空原内容）
a+	以读和追加模式打开
rb	以二进制读模式打开
wb	以二进制写模式打开（参见w）
ab	以二进制追加模式打开（参见a）
rb+	以二进制读写模式打开（参见r+）
wb+	以二进制读写模式打开（参见w+）
ab+	以二进制读写模式打开（参见a+）

标准文件

- 当程序启动后，以下三种标准文件有效



S_{ource}

```
>>> newcName = input('Enter the name of new company: ')
Enter the name of new company: Chaolihai
>>> print(newcName)
Chaolihai
```

- **fp.close()**

- fp为文件对象
- 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
```

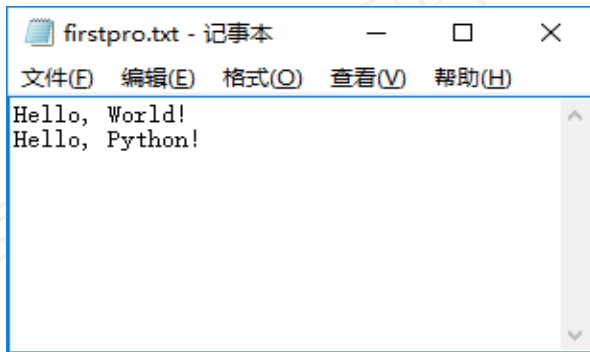
返回值和基本操作

- `open()`函数返回一个文件 (file) 对象
- 文件对象可迭代
- 有许多读写相关的方法/函数
 - `f.read()`, `f.write()`, `f.readline()`, `f.readlines()`, `f.writelines()`
 - `f.seek()`

读文件-read()方法

- **s = fp.read(size)**

- 从文件当前位置读取size字节数据，若size为负数或空，则读取到文件结束
- 返回一个字符串（文本文件）或字节流（二进制文件）




```
>>> 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)`**
 - 从文件当前位置读取本行内size字节数据，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

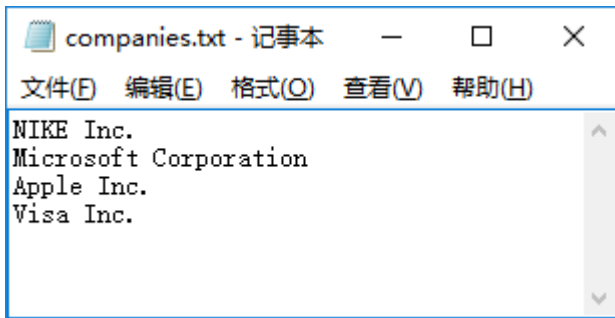
```
firstpro.txt :  
Hello, World!  
Hello, Python!
```

 `>>> 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)**

- 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

Source

```
>>> fp = open(r'd:\firstpro.txt', 'w')
>>> fp.write("Hello, World!\n")
14
>>> fp.write("Hello, Python!")
14
>>> fp.close()
```

Source

```
>>> f = open(r'd:\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

S
ource

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



```
try:
```

```
    with open(r'文件目录\test.txt') as fp:
```

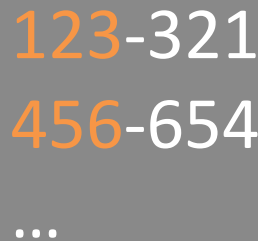
```
        ... # 各种文件处理
```

```
except IOError as err:
```

```
    print(err)
```

写入回文串

```
lst = []  
with open('c:/test/data1.txt') as fp:  
    for line in fp:  
        lineRev = line[::-1]  
        lst.append(line.strip()+'-'+lineRev.strip()+'\n')  
with open('data.txt', 'w') as fp:  
    fp.writelines(lst)
```



123-321
456-654
...

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。



Filename: prog1.py

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

Output:

```
1 GOOGLE Inc.
2 Microsoft Corporation
3 Apple Inc.
4 Facebook, Inc.
```

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、...后写到另一个文件scompanies.txt中。



Filename: prog2.py

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

Output:

```
1 GOOGLE Inc.  
2 Microsoft Corporation  
3 Apple Inc.  
4 Facebook, Inc.
```

文件的定位-`seek()`方法

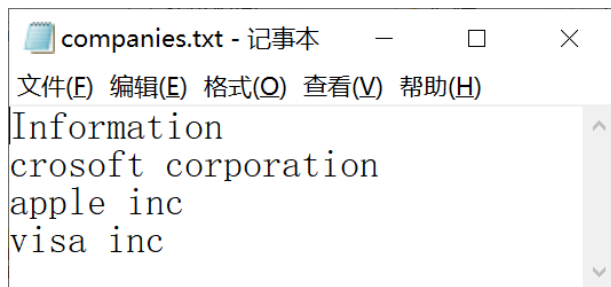
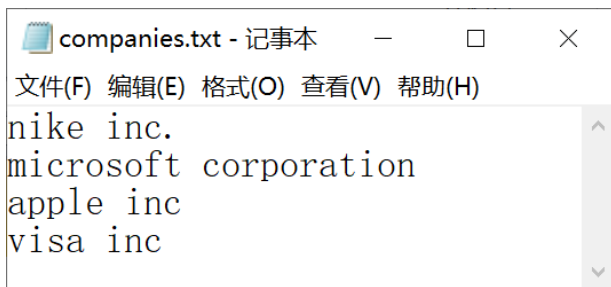
- `fp.seek(offset , whence=0)`
 - `fp`打开的文件必须允许随机访问
 - 在文件中移动文件指针，从`whence`（0表示文件头部，1表示当前位置，2表示文件尾部）偏移`offset`个字节
 - 返回当前的读写位置

文件中头部插入一个新行

F_{ile}

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

覆盖模式



统计某目录下的多个文本文件的行数

```
import os
```

```
def countLines(fname):
```

```
    with open(fname) as f:
```

```
        data = f.readlines()
```

```
    lens = len(data)
```

```
    print(fname + ' has ' + str(lens) + ' lines')
```

```
if __name__ == '__main__':
```

```
    # files = ['data1.txt', 'data2.txt', 'data3.txt', 'data4.txt']
```

```
    path = 'c:/test'
```

```
    files = os.listdir(path+'/data')
```

```
    for fname in files:
```

```
        countLines(path+'/data/'+fname)
```

```
if os.path.exists('./output'):  
    # 递归删除非空目录  
    shutil.rmtree('./output')  
os.mkdir('./output')
```

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