

目录

- [1. 文件的概念](#)
 - [1.1 文件的概念和作用](#)
 - [1.2 文件的存储方式](#)
- [2. 文件的基本操作](#)
 - [2.1 操作文件的套路](#)
 - [2.2 操作文件的函数/方法](#)
 - [2.3 read 方法 —— 读取文件](#)
 - [2.4 打开文件的方式](#)
 - [2.5 按行读取文件内容](#)
 - [2.6 文件读写案例 —— 复制文件](#)
- [3. 文件/目录的常用管理操作](#)
- [4. 文本文件的编码格式（科普）](#)
 - [4.1 ASCII 编码和 UNICODE 编码](#)
 - [4.2 Python 2.x 中如何使用中文](#)



1. 文件的概念

1.1 文件的概念和作用

- 计算机的 **文件**，就是存储在某种 **长期储存设备** 上的一段 **数据**
- 长期存储设备包括：硬盘、U 盘、移动硬盘、光盘…

文件的作用：

- 将数据长期保存下来，在需要的时候使用

CPU	内存	硬盘
		

1.2 文件的存储方式

- 在计算机中，文件是以 **二进制** 的方式保存在磁盘上的

文本文件和二进制文件：

- **文本文件**
 - 可以使用 **文本编辑软件** 查看
 - 本质上还是二进制文件
 - 例如：python 的源程序
- **二进制文件**
 - 保存的内容 不是给人直接阅读的，而是 **提供给其他软件使用的**
 - 例如：图片文件、音频文件、视频文件等等
 - 二进制文件不能使用 **文本编辑软件** 查看

2. 文件的基本操作

2.1 操作文件的套路

在 **计算机** 中要操作文件的套路非常固定，一共包含**三个步骤**：

- 1.打开文件
- 2.读、写文件
 - **读** 将文件内容读入内存
 - **写** 将内存内容写入文件
- 3.关闭文件

2.2 操作文件的函数/方法

在 Python 中要操作文件需要记住 1 个函数和 3 个方法

函数/方法	说明
open	打开文件，并且返回文件操作对象
read	将文件内容读取到内存
write	将指定内容写入文件
close	关闭文件

- `open` 函数负责打开文件，并且返回文件对象
- `read/write/close` 三个方法都需要通过 **文件对象** 来调用

2.3 read 方法 —— 读取文件

- `open` 函数的第一个参数是要打开的文件名（文件名区分大小写）
 - 如果文件 **存在**，返回 **文件操作对象**
 - 如果文件 **不存在**，会 **抛出异常**

- `read` 方法可以一次性 **读入** 并 **返回** 文件的 所有内容
- `close` 方法负责 **关闭文件**
 - 如果 **忘记关闭文件，会造成系统资源消耗，而且会影响到后续对文件的访问**
- **注意**：read 方法执行后，会把 **文件指针** 移动到 **文件的末尾**

```
# 1. 打开 - 文件名需要注意大小写
file = open("README")

# 2. 读取
text = file.read()
print(text)

# 3. 关闭
file.close()
```

提示：在开发中，通常会先编写 **打开** 和 **关闭** 的代码，再编写中间针对文件的 **读/写** 操作！

文件指针（知道）：

- **文件指针** 标记 **从哪个位置开始读取数据**
- **第一次打开** 文件时，通常 **文件指针会指向文件的开始位置**
- 当执行了 read 方法后，**文件指针** 会移动到 **读取内容的末尾**
 - 默认情况下会移动到 **文件末尾**

思考：如果执行了一次 read 方法，读取了所有内容，那么再次调用 read 方法，还能够获得内容吗？

答案：不能；第一次读取之后，**文件指针**移动到了文件末尾，再次调用不会读取到任何的内容

2.4 打开文件的方式

`open` 函数默认以 **只读方式** 打开文件，并且返回文件对象

语法如下：

```
f = open("文件名", "访问方式")
```

访问方式	说明
r	以只读方式打开文件。文件的指针将会放在文件的开头，这是默认模式。如果文件不存在，抛出异常
w	以只写方式打开文件。如果文件存在会被覆盖。如果文件不存在，创建新文件
a	以追加方式打开文件。如果该文件已存在，文件指针将会放在文件的结尾。如果文件不存在，创建新文件进行写入
r+	以读写方式打开文件。文件的指针将会放在文件的开头。如果文件不存在，抛出异常
w+	以读写方式打开文件。如果文件存在会被覆盖。如果文件不存在，创建新文件
a+	以读写方式打开文件。如果该文件已存在，文件指针将会放在文件的结尾。如果文件不存在，创建新文件进行写入

提示：频繁的移动文件指针，会影响文件的读写效率，开发中更多的时候会以 只读、只写 的方式来操作文件

写入文件示例：

```
# 打开文件
f = open("README", "w")

f.write("hello python! \n")
f.write("今天天气真好")

# 关闭文件
f.close()
```

2.5 按行读取文件内容

- read 方法默认会把文件的 所有内容 一次性读取到内存
- 如果文件太大，对内存的占用会非常严重

readline 方法

- readline 方法可以一次读取一行内容
- 方法执行后，会把 文件指针 移动到下一行，准备再次读取

读取大文件的正确姿势

```
# 打开文件
file = open("README")

while True:
    # 读取一行内容
    text = file.readline()
```

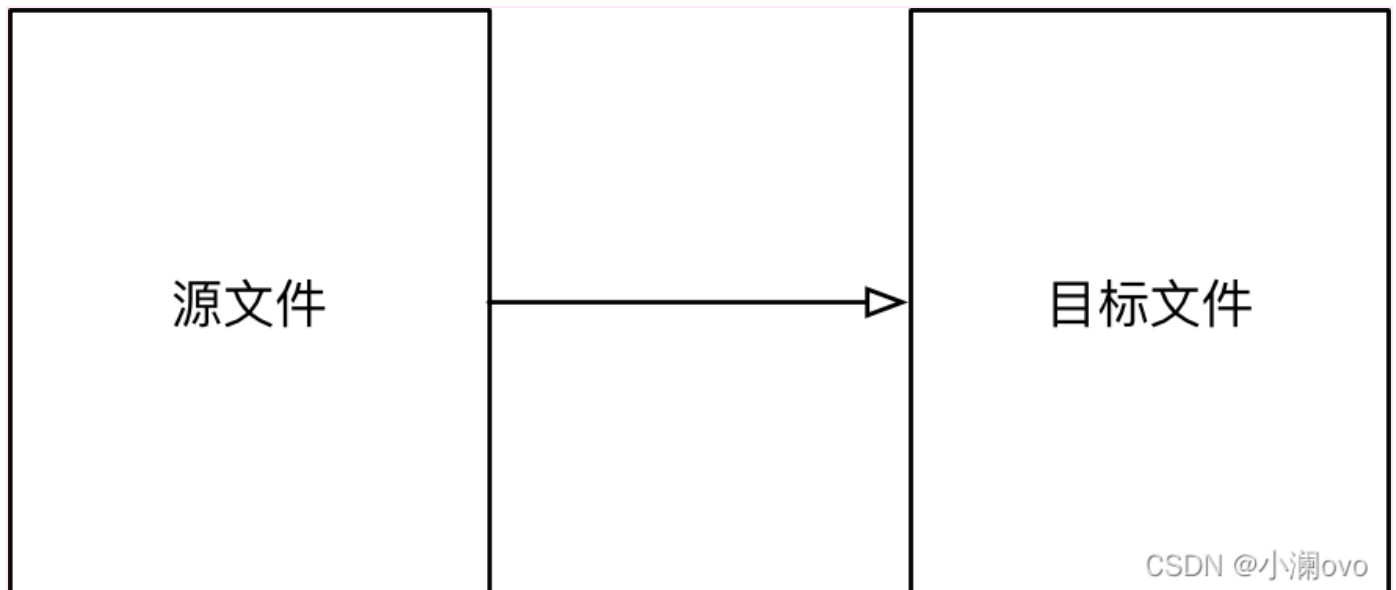
```
# 判断是否读到内容
if not text:
    break

# 每读取一行的末尾已经有了一个 `\\n`
print(text, end="")

# 关闭文件
file.close()
```

2.6 文件读写案例 —— 复制文件

目标：用代码的方式，来实现文件复制过程



小文件复制：打开一个已有文件，读取完整内容，并写入到另外一个文件

```
# 1. 打开文件
file_read = open("README")
file_write = open("README[复件]", "w")

# 2. 读取并写入文件
text = file_read.read()
file_write.write(text)

# 3. 关闭文件
file_read.close()
file_write.close()
```

大文件复制：打开一个已有文件，逐行读取内容，并顺序写入到另外一个文件

```
# 1. 打开文件
file_read = open("README")
file_write = open("README[复件]", "w")
```

```
# 2. 读取并写入文件
while True:
    # 每次读取一行
    text = file_read.readline()

    # 判断是否读取到内容
    if not text:
        break

    file_write.write(text)

# 3. 关闭文件
file_read.close()
file_write.close()
```

3. 文件/目录的常用管理操作

- 在 **终端 / 文件浏览器**、中可以执行常规的文件 / 目录管理操作，例如：
 - 创建、重命名、删除、改变路径、查看目录内容、……
- 在 Python 中，如果希望通过程序实现上述功能，需要导入 `os` 模块

文件操作：

方法名	说明	示例
rename	重命名文件	os.rename(源文件名, 目标文件名)
remove	删除文件	os.remove(文件名)
目录操作：		
方法名	说明	示例
-	-	-
listdir	目录列表	os.listdir(目录名)
makedirs	创建目录	os.makedirs(目录名)
rmdir	删除目录	os.rmdir(目录名)
getcwd	获取当前目录	os.getcwd()
chdir	修改工作目录	os.chdir(目标目录)
path.isdir	判断是否是文件	os.path.isdir(文件路径)

提示：文件或者目录操作都支持 **相对路径** 和 **绝对路径**

4. 文本文件的编码格式（科普）

- 文本文件存储的内容是基于 **字符编码** 的文件，常见的编码有 ASCII 编码，UNICODE 编码等
 - Python 2.x 默认使用 ASCII 编码格式
 - Python 3.x 默认使用 UTF-8 编码格式

4.1 ASCII 编码和 UNICODE 编码

ASCII 编码

- 计算机中只有 256 个 ASCII 字符
- 一个 ASCII 在内存中占用 **1 个字节** 的空间
 - 8 个 0/1 的排列组合方式一共有 256 种，也就是 $2^{**} 8$

ASCII表																									
(American Standard Code for Information Interchange 美国标准信息交换代码)																									
高四位	ASCII控制字符												ASCII打印字符												
	0000						0001						0010	0011	0100	0101	0110	0111							
	0						1						2	3	4	5	6	7							
低四位	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0		^@	NUL	\0	空字符	16	►	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	☺	^A	SOH		标题开始	17	◄	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	☹	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	♥	^C	ETX		正文结束	19	!!	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t	
0101	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	♠	^F	ACK		肯定应答	22	—	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	•	^G	BEL	\a	响铃	23	↕	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	▢	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(56	8	72	H	88	X	104	h	120	x	
1001	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41)	57	9	73	I	89	Y	105	i	121	y	
1010	A	◉	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	♂	^K	VT	\v	纵向制表	27	←	^[ESC	\e	溢出	43	+	59	;	75	K	91	[107	k	123	{	
1100	C	♀	^L	FF	\f	换页	28	└	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124		
1101	D	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93]	109	m	125	}	
1110	E	🎵	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	🎵	^O	SI		移入	31	▼	^_	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	␣	^Backspace 代码: DEL

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。

2023/04/08 洞见

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。

C2018/04/08 閑ovo

UTF-8 编码格式

- 计算机中使用 **1~6 个字节** 来表示一个 UTF-8 字符，涵盖了 **地球上几乎所有地区的文字**
- 大多数汉字会使用 **3 个字节** 表示
- UTF-8 是 UNICODE 编码的一种编码格式