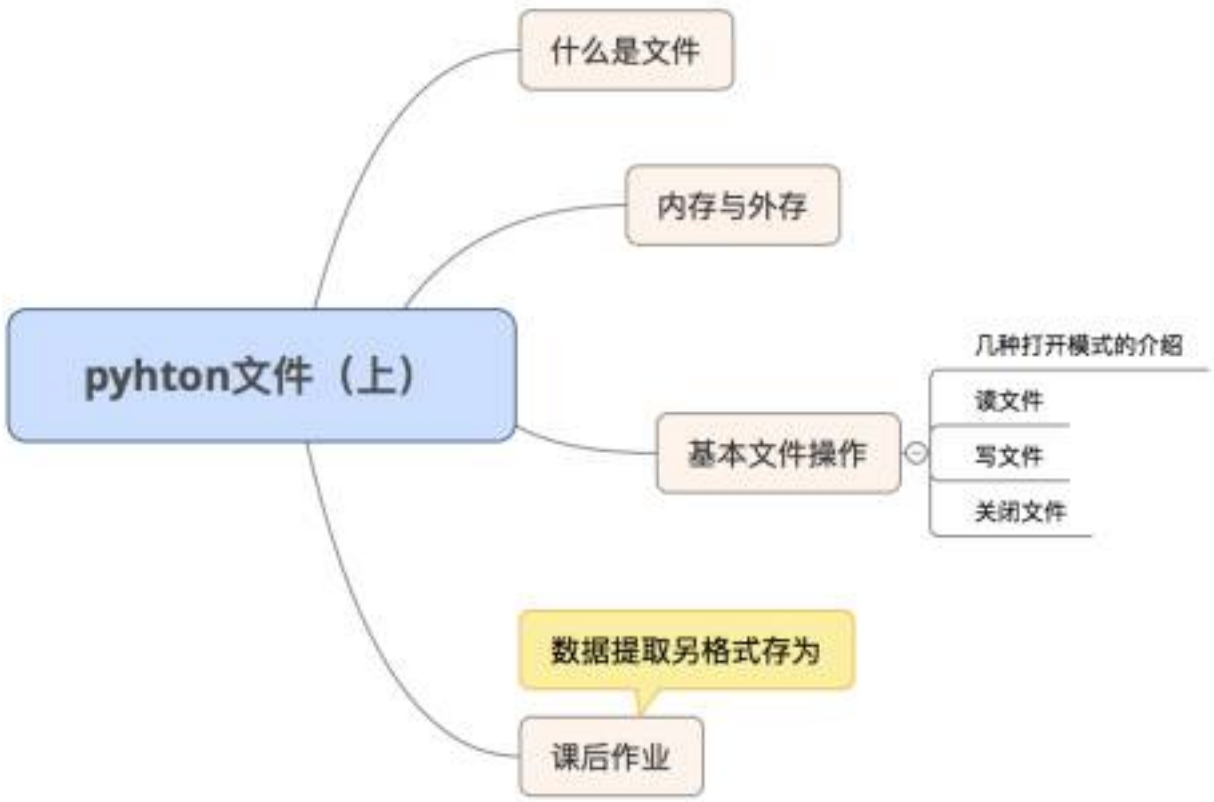
Python文件（上）输入、处理、输出

Original 2016-11-27 哈哈小 菜猿观世界

通过本节的学习，你将了解以下内容：



文件，不管是对计算机科班的，还是打酱油路过的，都不是一个陌生的词汇。

那么，什么是文件呢?

额呃，这个问题，没认真想过，好比我们一直是个人，却没想过怎样才是真真正正的人。

好吧，我们一起走走下文吧，希望哈哈小的浅识分享能让你们有所收获～

---

1 存储与文件

在进入主题之前，处于对计算机刚入门的同学的考虑，这里哈哈小要对计算机存储系统相关常识稍稍补充一下了。只想了解pyhton是如何操作文件的，请直达下一部分。

我们先将远程数据处理排除在外，不做考虑。在对计算机对数据的处理过程中，不管是现在哈哈小写原创的正在执行的浏览器，还是或者你正在边看python原创边玩的lol游戏，可以说，都包括数据的输入，处理和输出这三个过程。

计算机存储系统主要分两块，一是内存，我们一般正在执行的程序，都是把数据从外存即磁盘那搬到这来执行的。

比如某一个.py文件中有这么一片段：

```
for i in range(100):
    print(i)
```

这段代码在计算机底层执行的过程大概是这样的，这个.py文件写好之后先是保存在外存上，当要编译执行这段代码时，计算机操作系统收到指示，于是就把它搬运到内存上，按照一定的规则和逻辑，执行完毕，再通过操作系统把执行后的结果打印到显示屏上。最后我们就能看到那样的结果了。

内存大的特点是高速，数据处理速度可以上天，既然那么厉害，其价格就不言而喻了。虽然速度快，但不具备数据持久性，意思是一断电，数据就再也找不回来了，一关闭程序，就相当于不存在了。

比如我们运行某个基本的贪吃蛇游戏，玩到某个阶段显示的那个得分成绩就是存放在在内存中。所以当我们关闭程序，重新开始游戏时，那个分数又从零开始了。

这里肯定有的朋友会提出这样的问题，我们玩的游戏才没有那么低级，退出游戏之后，再次进入时，连个载入存档继续游戏的功能都没有吗?

很好，你的问题就是今天课时进行的必要性了。

其实，记录上次游戏状态的信息就是保存在外存中。

所以，学习编程，怎么能少得了跟文件打交道呢?

计算机存储系统主要分两块，刚才说了，第一个部分是内存。现在来说说第二块，也就是外存。

生活中，经常会出现这样一幕。

> 同学甲：你电脑内存多大啊?
>
> 同学乙：500GB

其实，同学乙回答的500GB并不是同学甲问的内存大小，而是我们现在讲的磁盘的大小。

外存的特点，就刚好与内存相反了，造价就显得特别“傲娇”，500GB的机械硬盘可能比4GB的内存还便宜，但是速度，与内存相比，就有点“低下头颅”的感觉了。

通过对比，就不难发现，事难两其美。就好比某些同学英语差一点，编程实力却很强，有些同学英语非常好，编程实力却一般。这就尴尬了。

不尴尬，既然这样，我们还有一件非常有必要做的事：做个折中，取长补短，物尽其美。

最终的结果就是，各有其用，皆大欢喜。

计算机就是不断伴随着这样的折中（trade-off）发展起来的。

讲到这里，文件是什么这个问题相对就好解释了，首先，想想我们都见过什么样的文件。

从文件名下手， xxxx.xxx 好像几乎每个文件都是这么个样子组成的 。

点号前面的xxxx就不用说了，我们平时说，给文件起个名字，说的就是这个xxxx啦。

点号后面的xxx叫做文件的拓展名，换句话说，就是决定这是一个什么样的文件的标记，即文件的类型的表明，再学术一点的话说，就是决定了文件内容是按照怎样的规则记录的标记。

比如我们常见的文件的类型有以下这些（随便举些例子）：

> .txt 基本文本文件
>
> .bmp 位图图像文件
>
> .avi这个应该很熟，是视频文件
>
> .mp3音频文件
>
> .exe Windows系统可执行文件

> .bat Windows系统批处理文件
>
> .sh Linux批处理文件

如果要在这对上面提到的学术一点的话做进一步的解释的话，就是一个这样的例子，为什么一个文本查看软件能打开.txt的文件，而不能正常打开.mp3这样的文件。

原因是，这个文本查看器不具备“通晓”.mp3这样文件规则的能力。相反 ， 音频播放软件就“懂”.mp3这样的文件（计算机术语就叫转码），因为“懂”，所以最后能让你听得见。

---

2 基本文件操作

基本点普及了，天也快亮了，该干点正事了，哈哈～

好吧，我们再回到python课堂，接下来就来探探python是怎样对文件操作的。

一个重要的函数－open，看看是哪来的神奇动物

```
>>> help(open)
Help on built-in function open in module io:
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None,
closefd=True, opener=None)
    Open file and return a stream.  Raise IOError upon failure.
```

（未完全显示）

通过函数文档可以得知，这个函数就是用来打开文件的。

这个函数可传入的参数好像还挺多的，我们今天只聚焦两个，稍提一个。

第一个是必填参数，file，其实就是你要打开文件的文件名

第二个是可选参数，如果不写，默认为r，好像不是太懂什么意思。

在这个open函数文档下面其实也有说明，由于是英文的，看着费劲，干脆哈哈小翻译过来，直接上中文的。

打开模式	执行操作
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。
w	打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
x	如果文件已存在，使用此方式打开会出异常
a	以写入模式打开，如果存在，则末尾追加写入
b	以二进制模式打开文件
t	可读写模式
U	通用换行符支持

这样就一见明朗了，原来，这里的mode就是打开模式的意思。

打开文件，我是只写呢？还是只读呢？还是又读又写呢？如果打开的是一个不存在的文件呢？等等这些问题，都是通过给定不同的打开模式做相应处理的。

第三个参数是可选参数，encoding，给定编码方式的意思，不指定默认为US-ASCII。这个解释就干脆上例子吧。

假如现在哈哈小有这么一文件：

文件名

> test.txt

文件内容

桑桑=183
哈哈小=182
黑炎龙的使者=178
秋词=170
桂焓=165

（有看过前面几篇教程的，应该能懂得文件内容的含义了，哈哈）

```
>>> f=open('test.txt',encoding="utf-8")
```

直接给文件名是因为文件就在当前目录下，如果不是的话，比如是放在c盘的根目录下，要给的文件路径应该是，c://test.txt。

接下来，哈哈小要展示的是对这个文件的读取操作，所以mode参数默认为r，只读就好了。

因为文件内容含中文，所以就给它个utf-8编码方式，这样去读取的时候就不会有任何的问题了。

f是什么？上面的函数文档中对open的描述是这样的

> Open file and return a stream

返回一个流对象，我们在这就把它叫做文件对象吧，既然有返回值，那我们就给个变量f去接受就好了。学过c的，也可以把这里的f近似看作c语言里的文件指针。

f这个文件对象有几个属性可以用用看

```
>>> f.name # 得到文件名
'test.txt'
>>> f.mode # 得到文件打开模式
'r'
>>> f.closed # 判断文件是否关闭
False
```

接下来，就到了这个文件对象所含方法的介绍了，也就是今天的核心，文件读写加关闭。

文件对象主要方法列举如下：

文件对象方法	执行操作
f.read(size=-1)	从文件读取指定的字节数，如果未给定或为负则读取所有。
f.readline()	读取整行，包括 "\n" 字符。
f.write(str)	将字符串写入文件，　有返回值。
f.writelines(seq)	向文件写入一个序列字符串列表，seq是一个返回字符串的可迭代对象。
f.seek(offset,from)	设置文件当前位置。
f.tell()	返回文件当前位置。
f.close()	关闭文件。关闭后文件不能再进行读写操作。

给出例子——解释

> f.read() 未给定任何值，默认为－1，读取所有

```
>>> f.read()
'桑桑＝183\n哈哈小＝182\n黑炎龙的使者＝178\n秋词＝170\n桂焓＝165'
```

> 这里的读取结果之所以为空，是因为自上步之后，文件当前位置已到尾端

```
>>> f.read()
''
```

> 不信的话，我们调用一下tell方法，让它告诉我们文件的当前位置

```
>>> f.tell()
69
```

> 确信无疑了，为了再次读取内容，我们需要调用seek方法，重新设置文件当前位置，seek有两个参数，表示设定到从from位置偏移offset个数的位置。from有三个数可给，0代表从头开始，1代表从当前位置开始，2表示从末位开始。比如，要让文件当前位置重回文件头，offset和from给的值就都是0。

```
>>> f.seek(0,0)
0
```

> 这时，如果我们再次调用read方法，与刚才不同，这此我们只读取前面10位，而不是全读

```
>>> f.read(10)
```

```
'桑桑=183\n哈哈小'
>>>
```

> 我们来看这一行还剩了些什么，剩些什么，就把这一行全部读取了吧。

```
>>> f.readline()
'=182\n'
>>>
```

> 我们重新归位，并做这样一个操作

```
>>> f.seek(0,0)
```

```
>>> list(f)
['桑桑=183\n', '哈哈小=182\n', '黑炎龙的使者=178\n', '秋词=170\n', '桂焓=165\n', '疏桐＝170']
```

神奇的地方在于，list具有传入函数对象直接按行依次迭代并分装成一个列表的能力。

> 其实，我们也可以直接用for循环去遍历这个集合对象，这过程大概就与list原理一样

```
>>> f.seek(0,0)
0
>>> for line in f:
        print(line)

桑桑=183
哈哈小=182
黑炎龙的使者=178
秋词=170
桂焓=165
>>>
```

> 读取操作完毕，记住，要养随开随关的好习惯

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

```
>>> f.closed
True # 这里可知，文件已经关闭了
```

读取完毕，接下来我们要对同一个文件进行写操作。现在要将一条新的数据疏桐＝170置于文件的最后

> 由于我们将要以追加文件末尾的方式操作文件，所以我们以a（append的首字母）模式打开这个文件

```
>>> f=open('/Users/wangcongcong/Desktop/test.txt','a',encoding="utf-8")
```

> 写入，有发现这里传给writelines方法的是一个列表，也就是一个的迭代对象。如果要插入两行这样的数据的话，参数可以这样写：["\n疏桐＝170"，"\n之户先生＝176"]

```
>>> f.writelines(["\n疏桐＝170"])
```

> 我们f.close()之后，再去查看文件，发现内容已被写进

> test.txt

```
桑桑=183
哈哈小=182
黑炎龙的使者=178
秋词=170
桂焓=165
疏桐＝170
```

write方法就更灵活了，想写入什么，就直接给字符串就好了，还有一点不同在于，f.write(str)执行完成之后，将会返回写入后文件所在的位置值。

---

3 课后作业

今天的课后作业就简单了，问题的描述是这样的：

上文操作的test.txt文件内容已经是这样的了

> test.txt

桑桑=183
哈哈小=182
黑炎龙的使者=178
秋词=170
桂焓=165
疏桐=170

现在要求将该文件重构成如下样子：

> test.json

{"桑桑":183
,"哈哈小":182
,"黑炎龙的使者":178
,"秋词":170
,"桂焓":165
,"疏桐":170}

有发现，要完成此题，有两点要注意，一是文件格式即文件后缀名由txt变为json，二是文件内容键值对的格式稍有改变了。

开动脑筋，哼哼哈嘿，去想像吧～

对了，想要答案找哈哈小就是了。

结束语：我们不想给你带来多少多少的知识，我们只想尽绵薄之力给你带来可能的智慧的启迪。