Python 文件操作

FILES OPERATION USING PYTHON

之文件读写

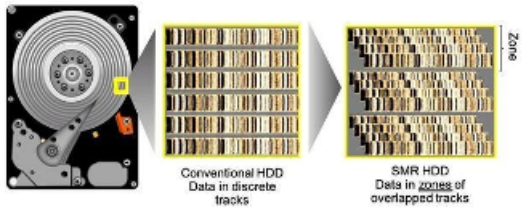
READ OR WRITE

01 文件的概念

文件的概念

文件是存储在辅助存储器上的数据集合

通常可以长久保存，也称为磁盘文件。



文件的类型

文 本 文 件

（1）基于字符编码，文件的内容就是字符

（2）存取是以字符为单位的，输入/出字符流的开始和结束由程序控制。

（3）用通用的记事本就可以浏览，具有可读性，因此，在存取时需要编码/解码，从而花费一定的转换时间。

二进制文件

（1）二进制文件直接由0，1组成，没有统一字符编码

（2）数据按照其实际占用的字节数存放

（3）不需要编/解码,不存在转换时间，但通常无法直接读懂

python 3.X 环境中：

文件的默认编码是utf-8 ， 字符串使用的编码是 Unicode。

文件的概念

文件的访问

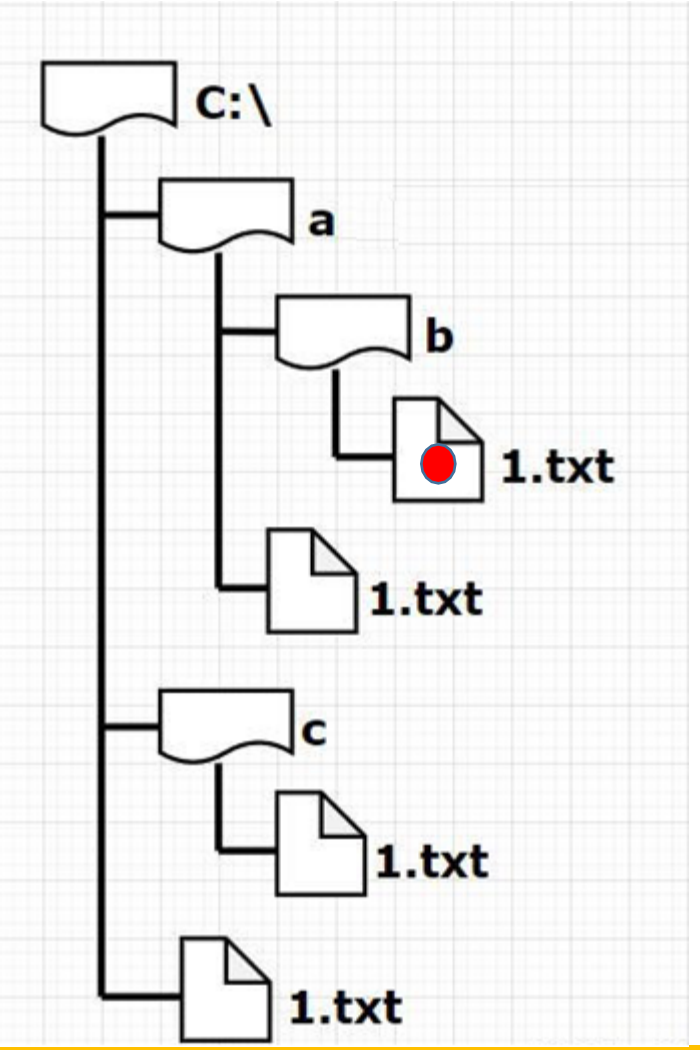
绝对路径：

- 如：c:\a\b\1.txt
- 从根目录开始标识文件所在的完整路径的方式。
- 同个文件，路径相同

相对路径：

- 如：b\1.txt
- 从所在位置开始标识，相对于程序所在目录位置建立
- 参照引用文件所在的路径。
- 同个文件，路径不同

多级目录结构



文件的概念

python操作文件整体过程与使用Word 软件 编写一份文件的过程相同：打开文件或者新建立一个文件，读/写数据，关闭文件。

用 Python 内置的 open( )函数打开一个文件。

fileobject = open(file_name [, access_mode][, buffering])

- fileobject：文件对象
- file_name：文件路径和名称
- access_mode：文件打开模式
- buffering：
  - <0: 默认为-1
  - =0:关闭缓存区
  - =1:缓存文件的行
  - >1:缓存大小

如果要读写一个文件，首先要建立一个文件对象，再利用文件对象提供的方法对文件的数据进行读写操作。

文件的概念

文件打开模式——打开模式指定了打开文件后处理方式

模式	描述
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
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式
r+	打开一个文件用于读写。文件指针将会放在文件的开头
rb+	以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头
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，则创建新文件
wb	以二进制格式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，则创建新文件
w+	打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，则创建新文件

文件的概念

模式	描述
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，则 创建新文件
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的 内容将会被写入到已有内容之后。如果该文件不存在，则创建新文件进行写入
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也 就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，则创建新文件进行写入
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾，文件打开时采用 的是追加模式。如果该文件不存在，则创建新文件用于读写
ab+	以二进制格式打开一个文件用于读写追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，则创建新文件用于读写

小结

01 **文件的类型**

文本文件

二进制文件

02 **文件的访问**

绝对路径

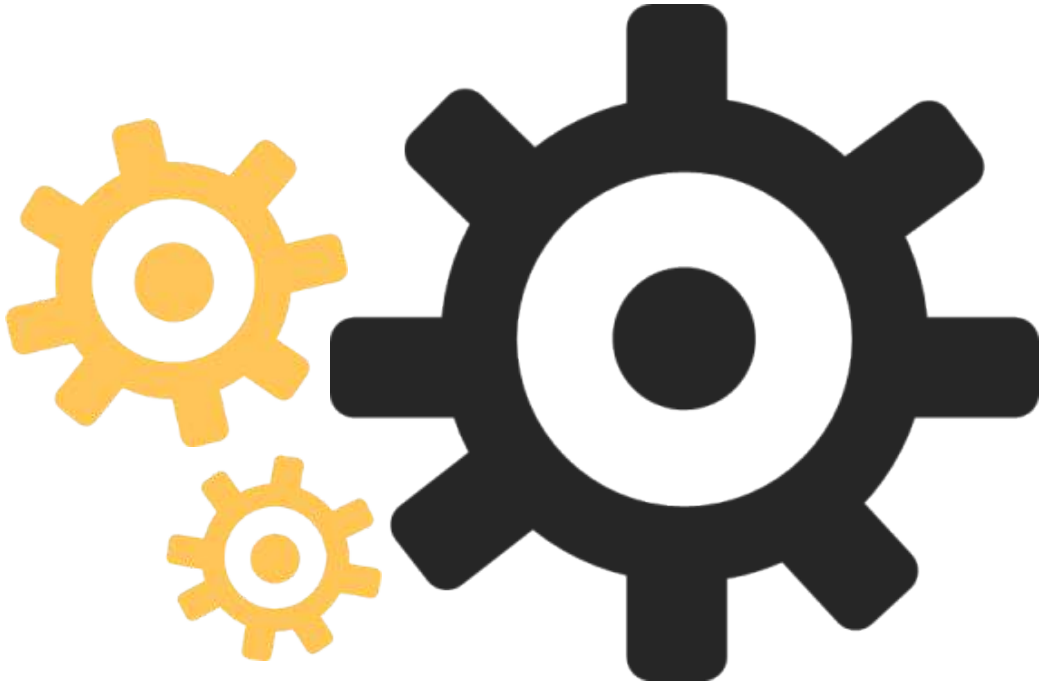
相对路径

03 **文件的打开模式**

读模式

写模式

追加模式



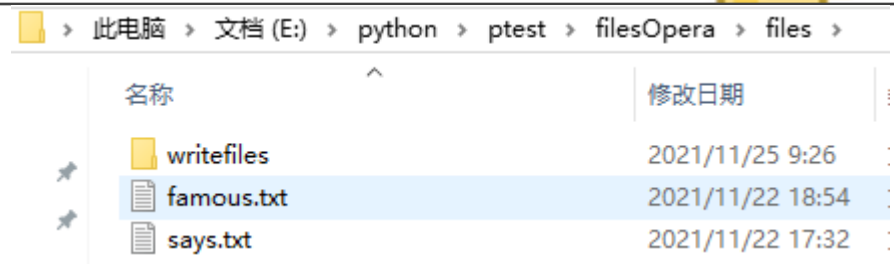
02 文件的打开与关闭

文件的打开与关闭

fileobject = open(file_name [, access_mode][, buffering])

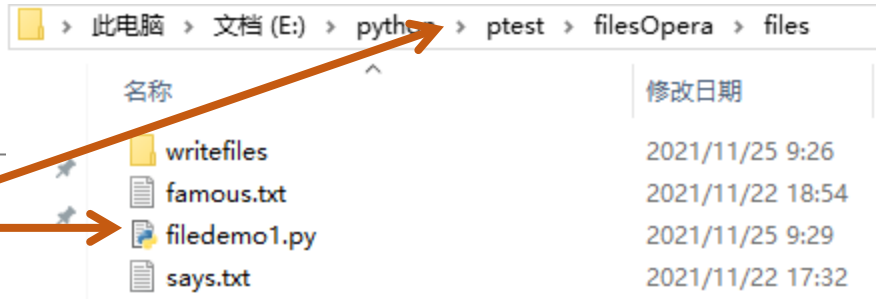
属性	描述
closed	如果文件已被关闭返回 True，否则返回 False
mode	返回被打开文件的访问模式
name	返回文件的名称
buffer	返回当前文件的缓冲区对象

打开文件



例1：设当前目录： E:\python\ptest\filesOpera\files，下有两个文本文件：famous.txt和says.txt，一个空文件夹 writefiles

1.在当前目录下编写程序filedemo1.py，以读模式打开与源程序处于同一目录 的says.txt 文件




```
f1=open(“says.txt”,”r”)
```

等价于：

```
f1=open("E:/python/ptest/filesOpera/files/says.txt","r")
```

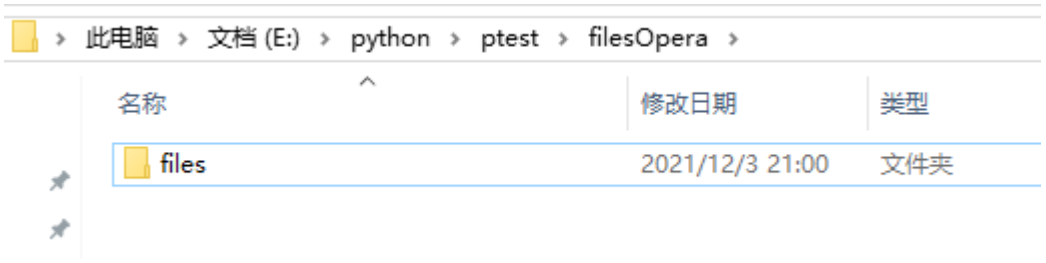
打开文件

 # 2.在上一级目录下编写程序filedemo2.py，以二进制读模式打开处于源程序下一级目录 的says.txt 文件

```
f2=open("files/says.txt","rb")
```

等 价于：

```
f2=open("E:/python/ptest/filesOpera/files/says.txt","r")
```



打开文件

 # 3.writefiles下编写程序filedemo3.py，以写模式打开上一级目录 的hello.txt 文件

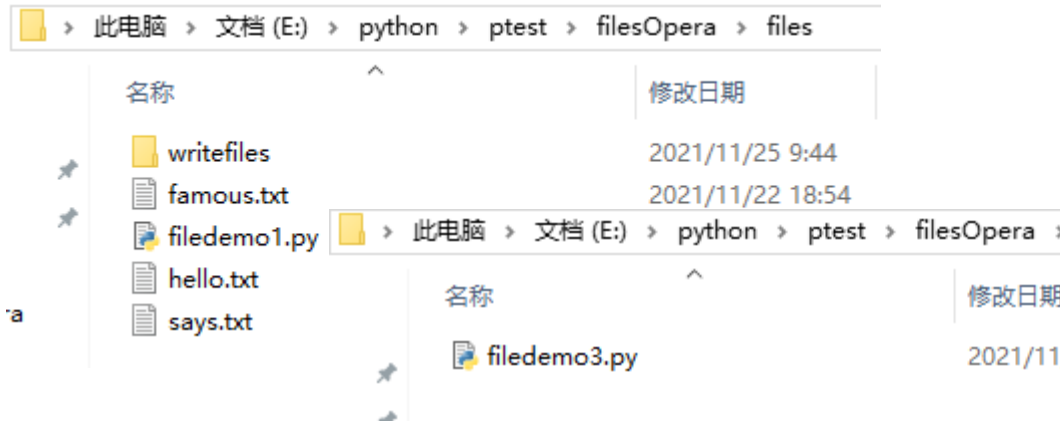
```
f3=open("../hello.txt","w")
```

等 价于：

```
f3=open("E:/python/ptest/filesOpera/files/hello.txt","w")
```

文件不存在，则创建新文件

文件存在，则覆盖原文件



关闭文件

文件打开状态下才能对文件进行读或取操作

同样：

文件关闭才能将数据写入磁盘

利用 python内置函数 close()来关闭文件

如：**fileobject.close()**

filedemo3.py - E:/python/ptest/filesOpera/files/writefiles/filedemo3.py (3.7.0)

File Edit Format Run Options Window Help

```
f3=open("../hello.txt","w")
print(f3)
f3.close()
```

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

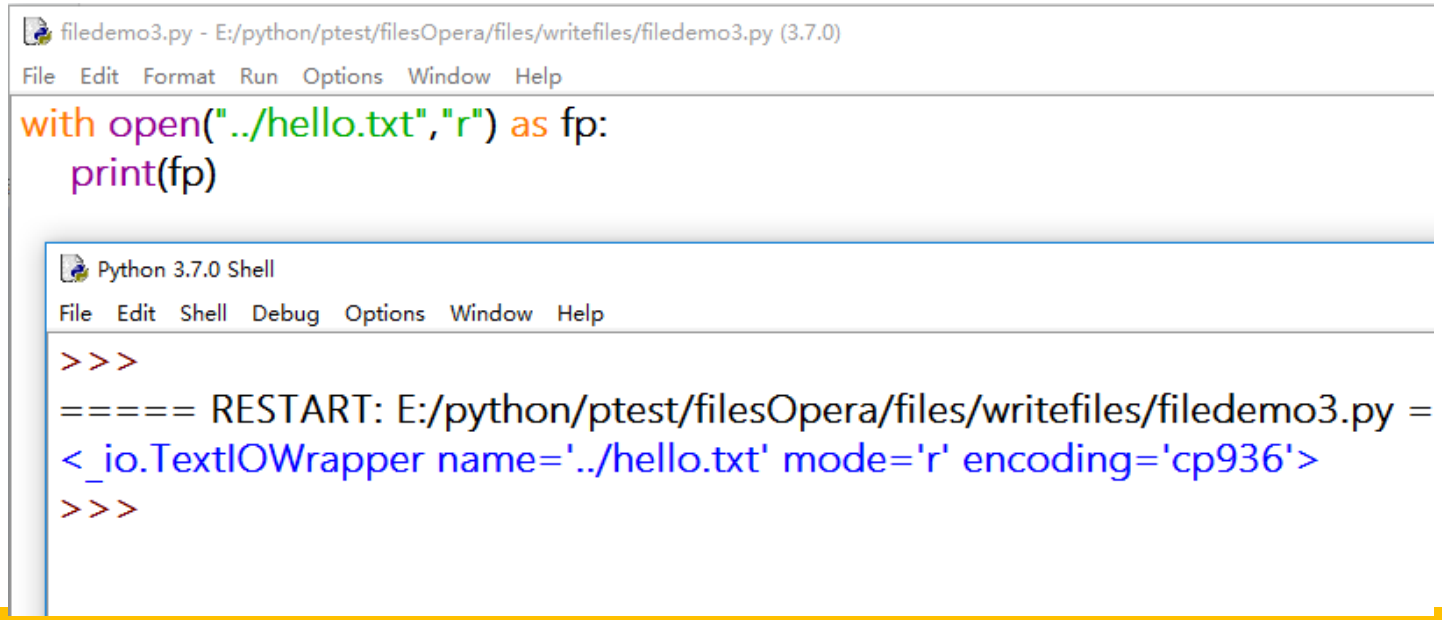
```
>>>
===== RESTART: E:/python/ptest/filesOpera/files/writefiles/filedemo3.py =
<_io.TextIOWrapper name='../hello.txt' mode='r' encoding='cp936'>
>>>
```

关闭文件

需要注意的是，即使写了关闭文件的代码，也无法保证文件一定能够正常关闭。例如，如果在打开文件后和关闭文件之前发生了错误，导致程序崩溃，这时文件就无法正常关闭。在管理文件对象时推荐使用with 关键字，可以有效避免这个问题。

with open(filename,mode) as fp:

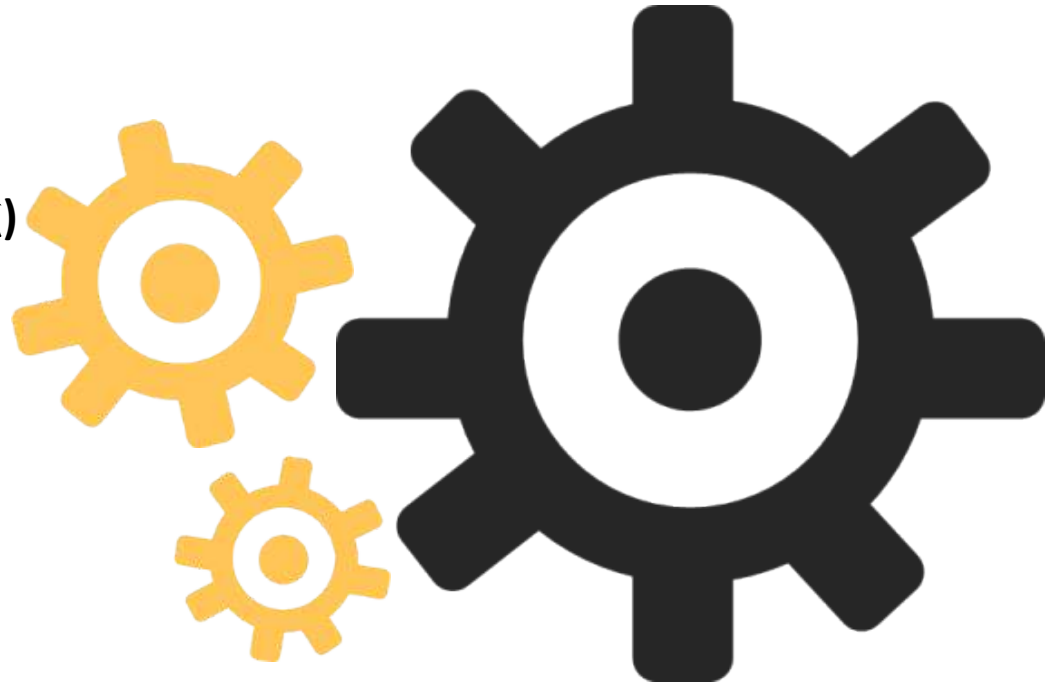
文件对象操作语句；



小结

01 打开文件

fileobject=open(文件路径,读写模式)



02 关闭文件

fileobject.close()

或

with 语句

03 文件的读写

读文件

格 式

```
fileobject.read()
fileobject.read(n)
fileobject.readline(n)
fileobject.readlines()
```

- **read()**：一次性读入文件中所有内容。
- **read(n)**：读取指定的n 个字符；若n 大于文件长度，则读取所有内容。
- **readline(n)**：若n 没有，则读取一行，若n 有，则读取一行中的n 个字符。
- **Readlines()**：一次性将文件的所有内容全部读出，以列表形式返回。

  一般再次使用 for 循环从readlines()中提取每一行。