目录

[文件操作 1](#_Toc485366564)

[加载文件 1](#_Toc485366565)

[读文件 1](#_Toc485366566)

[写文件 1](#_Toc485366567)

[文件指针 2](#_Toc485366568)

[获取文件属性的其他方法 2](#_Toc485366569)

[使用指定文件编码格式创建和打开文件 2](#_Toc485366570)

[OS模块对文件的操作 2](#_Toc485366571)

[os.path的方法 3](#_Toc485366572)

[数据库操作 3](#_Toc485366573)

[操作MySql数据库 3](#_Toc485366574)

[游标对象 cursor 3](#_Toc485366575)

[create 4](#_Toc485366576)

[insert / delete / update 4](#_Toc485366577)

[query 5](#_Toc485366578)

# 文件操作

## 加载文件

open(name,mode,buf) name 文件路径; mode 打开方式,不加为只读; buf 缓冲buffering大小

**mode:**

'r'只读 : 若以只读方式'r'打开,文件必须存在

'w'只写 : 以只写方式'w'打开,当文件不存在创建当前文件,若文件存在会将文件清空

'a'追加 : 以追加方式打开文件不存在时会创建文件

'r+'或'w+'可读可写 : 以'r+'方式打开文件写入会从头写入,将之前的内容覆盖写入的长度;

以'w+'方式打开文件还是会将文件清空

'a+' 追加和读写

每个字母后面都可以加上b,表示用二进制方式打开,比如 'rb+'.

比如: a = open('test.txt', 'r+')

## 读文件

b = a.read()

读取文件,可以传入一个参数size,读取size个字节,默认为全部

b = a.readline()

读取一行,可以传入一个参数size.若行的长度小于size会返回整行,若行的长度大于size会返回size个长度,继续readline(size)会往后继续读  
  
b = a.readlines()  
读取完文件,返回每一行组成的列表(若文件很大会占用很多内存空间,不推荐)  
可以传入一个参数size,这个size表示每次只读取大小为io.DEFAULT\_BUFFER\_SIZE \* size相接近的大小的字节.  
  
iter\_d = iter(a)  
使用迭代器对文件进行迭代,接着即可用for循环迭代,迭代器不会将文件全部加载到内存,只会在调用next时会读取一行.

## 写文件

a.write('123') 将字符串写入文件

write会有一个缓冲区,当调用close()或flush()或写入的数据量大于或等于缓冲区大小时会将缓存同步到磁盘,在写文件后需要调用close()或flush(),将缓冲区的内容写到文件,若不调用可能会丢失内容.

a.writelines(('12', '12', '12')) 将多行写入文件,参数必须为字符串组成的列表或元组

## 文件指针

a.seek(offset, whence)

offset: 偏移量,可以为负数,正数表示往后,负数表示往前,当文件的偏移大于了文件的长度会报错;

whence: 偏移相对位置

偏移相对位置:

os.SEEK\_SET 相对于文件起始位置,0  
os.SEEK\_CUR 相对于文件当前位置,1  
os.SEEK\_END 相对于文件结束位置,2

a.tell() 可以返回当前文件的偏移

比如:

a.seek(0, os.SEEK\_SET) 回到文件起始位置  
a.seek(-1, os.SEEK\_CUR) 表示在当前位置向前移动一位

## 获取文件属性的其他方法

file.fileno() 返回文件描述符号

file.mode() 返回文件打开权限

file.closed() 返回文件是否关闭

file.encoding() 3.x才有,返回文件编码

## 使用指定文件编码格式创建和打开文件

使用codecs模块的方法:

codecs.open('文件名称', '打开方式' , '指定编码的格式') 打开方式默认为rb

## OS模块对文件的操作

os.open(filename,flag) flag: os.O\_CREAT 创建文件 os.O\_RDONLY 只读

os.O\_WRONLY 只写 os.O\_RDWR 可读可写

os.read(fd,buffersize) 读取(fd表示文件对象,下同)

os.write(fd,string) 写入

os.lseek(fd,pos,how) 文件指针操作. 比如: os.lseek(a, 0, os.SEEK\_SET)

os.close(fd) 关闭文件

os.access(path,mode) 判断该文件权限. (path表示文件路径,下同)

mode: F\_OK 存在, R\_OK 读权限, W\_OK写权限, X\_OK 执行权限

os.listdir(path) 返回当前目录下所有文件组成的列表

os.remove(path) 删除文件,文件不存在时会报错

os.rename(old,new) 修改文件或者目录名,文件或目录不存在时会报错.

os.mkdir(path) 创建目录

os.makedirs(path) 创建多级目录,比如: os.makedirs('dir/test/test2')

os.removvedirs(path) 删除多级目录,多级目录中必须存在并且没有文件,否则会报错

os.rmdir(path) 删除目录,目录必须空目录

### os.path的方法

os.path.exists(path) 当前路径是否存在

os.path.isdir(fileName) 是否是一个目录

os.path.isfile(path) 是否是一个文件

os.path.getsize(filename) 返回文件大小,单位为字节

os.path.dirname(path) 返回路径的目录,比如 'pack/Class.py' 返回pack

os.path.basename(path) 返回路径的文件名,比如 'pack/Class.py' 返回Class.py

# 数据库操作

## 操作MySql数据库

下载MySQLdb或pymysql,其中MySQLdb不支持Python3.x,pymysql支持2.7和3.x,两者使用的方法基本完全一样. pip install PyMySQL / MySQLdb

连接对象:建立Python客户端与数据库的网络连接,返回一个Connection对象:

db = MySQLdb / pymusql.connect(host="127.0.0.1", user="root", passwd="123", db="test", charset="utf8")

host 服务器地址 port 端口号,默认3306 user 用户名 passwd 密码

db 数据库名称 charset 连接编码(utf8)

Connection对象支持的方法:

cursor = db.cursor() # 创建连接并返回游标

db.commit() # 提交当前事务

db.rollback() # 回滚当前事务

db.close() # 关闭连接

### 游标对象 cursor

cursor.execute(sql) # 执行一个sql语句

result = cursor.fetchone() # 取结果集下一行,返回字段顺序的元组

results = cursor.fetchmany(size) # 取结果即下size行,返回存储多行的元组

results = cursor.fetchall() # 获取结果集剩下的所有行,返回存储多行的元组

cursor.close() # 关闭游标

cursor.rowcount # 最近一次执行execute返回数据的行数或影响的行数

使用execute执行的sql语句为标准sql语句.

# coding=utf-8  
**import** MySQLdb  
  
db = MySQLdb.connect(host="127.0.0.1", port=3306, user="root", passwd="123", db="test", charset="utf8")  
cursor = db.cursor()

# 查询打印数据库版本  
cursor.execute("SELECT VERSION()")  
data = cursor.fetchone()  
**print** "Database version : %s " % data

## create

# 如果数据表已经存在使用 execute() 方法删除表。  
cursor.execute("DROP TABLE IF EXISTS test")  
  
# 创建数据表SQL语句  
sql = """CREATE TABLE test(  
 FIRST\_NAME **CHAR**(20) NOT NULL,  
 LAST\_NAME **CHAR**(20),  
 AGE **INT**,  
 SEX **CHAR**(1),  
 INCOME **FLOAT** )"""  
cursor.execute(sql)

## insert / delete / update

sql = """INSERT INTO test(FIRST\_NAME, LAST\_NAME, AGE, SEX, INCOME)   
 VALUES ('%s', '%s', '%d', '%c', '%d' )""" \  
 % ('23123', '1231231', 101231, 'M', 20001)

sql = "DELETE FROM test WHERE AGE > '%d'" % (20)

sql = "UPDATE test SET AGE = AGE + 1 WHERE SEX = '%c'" % 'M'

# 增删改只是sql语句的区别.  
**try**:  
 cursor.execute(sql)   
 db.commit()  
**except**:  
 db.rollback() # 如果发生错误,回滚数据

cursor.close()  
db.close()

## query

sql = 'select \* from test where INCOME > 1000'  
cursor.execute(sql)  
  
# 读取一行时  
result = cursor.fetchone()  
**print** result[0], result[1], result[2], result[3], result[4]  
  
# 读取剩余所有行  
results = cursor.fetchall()  
**for** row **in** results:   
 fname = row[0]  
 lname = row[1]  
 age = row[2]  
 sex = row[3]  
 income = row[4]  
 **print** fname, lname, age, sex, income  
  
cursor.close()  
db.close()