文件与目录

milo

目标

- 文件的打开和创建
- 文件读取
- 文件写入
- 内容查找和替换
- 文件删除,复制,重命名
- 目录操作

案例

- 目录分析器
- 杀毒软件
- 系统垃圾清理工具

Python文件读写

第一节

python文件读写

- python进行文件读写的函数是open或file
- file_handler = open(filename,,mode）

python文件读写

- mode

模式	说明
r	只读
r+	读写
w	写入，先删除原文件，在重新写入，如果文件没有则创建
w+	读写，先删除原文件，在重新写入，如果文件没有则创建（可以写入输出）
模式	说明
a	写入：在文件末尾追加新的内容,文件不存在，创建之
a+	读写：在文件末尾追加新的内容，文件不存在，创建之
b	打开二进制的文件。可以与r, w, a, +结合使用
U	支持所有的换行符号。“\r”, “\n”, “\r\n”

文件对象方法

第二节

milo

文件对象方法

- 文件对象方法
  - FileObject.close()
  - String = FileObject.readline([size])
  - List = FileObject.readlines([size])
  - String = FileObject.read([size])
  - FileObject.next()
  - FileObject.write(string)
  - FileObject.writelines(List)
  - FileObject.seek(偏移量, 选项)
  - FileObject.flush()

文件对象方法

- close:
  - 格式：
    - FileObject.close()
    - 关闭打开的文件对象
  - NOTE：
    - 每次调用都需要使用

文件对象方法

- readline：
  - 格式
    - String = FileObject.readline([size])
  - 说明：
    - 每次读取文件的一行
    - size: 是指每行每次读取size个字节，直到行的末尾

文件对象方法

- readlines：
  - 格式：
    - List = FileObject.readlines([size])
  - 说明：
    - 多行读，返回一个列表
    - size: 每次读入size个字符，然后继续按size读， 而不是每次读入行的size个字符

文件对象方法

- read：
  - 格式：
    - String = FileObject.read([size])
  - 说明：
    - 读出文件的所用内容，并复制给一个字符串
    - size: 读出文件的前 [size]个字符, 并输出给字符串，此时文件的指针指向size处

文件对象方法

- next：
  - 格式：
    - FileObject.next()
  - 说明：
    - 返回当前行，并将文件指针到下一行

文件对象方法

- write：
  - 格式：
    - FileObject.write(string)
  - 说明：
    - wirte 和后面的writelines在写入前会是否清除文件中原来所有的数据，在重新写入新的内容，取决于打开文件的模式

文件对象方法

- writelines：
  - 格式：
    - FileObject.writelines(List)
  - 说明：
    - 多行写
    - 效率比write高， 速度更快， 少量写入可以使用write

文件对象方法

- FileObject.seek(偏移量, 选项)
  - 选项 =0 时， 表示将文件指针指向从文件头部到 "偏移量"字节处。
  - 选项 =1 时， 表示将文件指针指向从文件的当前位置，向后移动 "偏移量"字节。
  - 选项 =2 时， 表示将文件指针指向从文件的尾部，，向前移动 "偏移量"字节。

文件对象方法

- FileObject.flush()
  - 提交更新

文件查找和替换

- 文件查找
- cat a.t

hello world

hello hello world

统计文件中hello的个数

python文件读写

- 示例：

```
import re
fp = file("a.t”, "r")
count = 0
for s in fp.readlines():
        li = re.findall("hello”, s)
        if len(li) > 0:
                count = count + len(li)
print “Search  ” + count + “   hello”
fp.close()
```

python文件读写

- 文件内容替换
- 问题： 把a.t中的hello 替换为 csvt, 并保存结果到文件a2.t中

python文件读写

- 示例一：
  - fp1 = file("a.t", "r")
  - fp2 = file("a2.t”, "w")
  - for s in f1.readlines():
  - fp2.write(s.replace("hello”, “csvt"))
  - fp1.close()
  - fp2.close()

python文件读写

- 示例二：
  - fp1 = file("a.t", "r+")
  - s = f1.read():
  - f1.seek(0,0)
  - f1.write(s.replace("hello”, “csvt"))
  - fp1.close()

OS模块

第一节

milo

目录操作

- 目录操作就是通过python 来实现目录的创建，修改，遍历等功能
- import os
  - 目录操作需要调用os模块
  - 比如：
    - os.mkdir('/root/csvt')

常用函数

函数	说明
mkdir(path[,mode=0777])	
makedirs(name, mode=511)	
rmdir(path)	
removedirs(path)	
listdir(path)	
getcwd()	
chdir(path)	
walk(top,topdown=True, onerror=None)	

目录遍历-milo

- 案例
  - 系统垃圾清除小工具
- 方式：
  - 递归函数
  - Os.walk()函数

```
# 递归遍历目录
import os
def VisitDir(path):
    li = os.listdir(path)
   for p in li:
       pathname = os.path.join(path, p)
       if not os.path.isfile(pathname):
           VisitDir(pathname)
       else:
           print pathname

if __name__ == "__main__":
   path = r“/test"
   VisitDir(path)
```

目录遍历

- os.walk()
  - 函数声明：os.walk(path)
- 该函数返回一个元组，该元组有3个元素，这3个元素分别表示每次遍历的路径名，目录列表和文件列表

目录遍历

- os,walk()实例:

```
import os
def VisitDir(path):
  for root,dirs,files in os.walk(path):
    for filespath in files:
      print os.path.join(root,filespath)
if __name__=="__main__":
path="/root"
VisitDir(path)
```

python文件读写

小结

- ;

总结

- 掌握Python文件操作的方法