文件操作

文件操作基本函数

open()函数概述

Python open() 方法用于打开一个文件，并返回文件对象，在对文件进行处理过程都需要使用到这个函数，如果该文件无法被打开，会抛出 OSError。

注意：使用 open() 方法一定要保证关闭文件对象，即调用 close() 方法。

open() 函数常用形式是接收两个参数：文件名(file)和模式(mode)。打开文件的操作如下所示。

```
open(file, mode='r')
```

打开文件的常用模式

- 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。
- r+ 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打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，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打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，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，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。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。

文件操作常用的函数

close()函数的用法

close() 方法用于关闭一个已打开的文件。关闭后的文件不能再进行读写操作， 否则会触发 ValueError 错误。 close() 方法允许调用多次。

当 file 对象，被引用到操作另外一个文件时，Python 会自动关闭之前的 file 对象。 使用 close() 方法关闭文件是一个好的习惯。

实例1:close()函数的使用

```
# 打开文件
fo = open("runoob.txt", "wb")
print("文件名为: ", fo.name)

# 关闭文件
fo.close()
```

输出结果：

```
文件名为:  runoob.txt
```

read()函数的用法

read() 方法用于从文件读取指定的字节数，如果未给定或为负数则读取所有。

实例1:read()函数的使用

假设runoob.txt文件内容如下：

```
这是第一行
这是第二行
这是第三行
这是第四行
这是第五行
```

实例代码：

```
# 打开文件
fo = open("runoob.txt", "r+")
print ("文件名为: ", fo.name)

line = fo.read(10)
print ("读取的字符串: %s" % (line))

# 关闭文件
fo.close()
```

输出结果：

```
文件名为:  runoob.txt
读取的字符串: 这是第一行
这是第二
```

readline()函数的使用

readline() 方法用于从文件读取整行，包括 "\n" 字符。如果指定了一个非负数的参数，则返回指定大小的字节数，包括 "\n" 字符。

实例1:readline()函数的使用

假设runoob.txt文件的内容如下：

```
1:www.runoob.com
2:www.runoob.com
3:www.runoob.com
4:www.runoob.com
5:www.runoob.com
```

实例代码：

```
# 打开文件
fo = open("runoob.txt", "r+")
print ("文件名为: ", fo.name)

line = fo.readline()
print ("读取第一行 %s" % (line))

# 传入size参数为5，则读出5个字符
line = fo.readline(5)
print ("读取的字符串为: %s" % (line))

# 关闭文件
fo.close()
```

输出结果：

```
文件名为:  runoob.txt
读取第一行 1:www.runoob.com

读取的字符串为: 2:www
```

readlines()函数的使用

readlines() 方法用于读取所有行(直到结束符 EOF)并返回列表，该列表可以由 Python 的 for... in ... 结构进行处理。 如果碰到结束符 EOF 则返回空字符串。
如果碰到结束符 EOF 则返回空字符串。

实例1:realines()函数的使用

假设runoob.txt文件内容如下：

```
1:www.runoob.com
2:www.runoob.com
3:www.runoob.com
4:www.runoob.com
5:www.runoob.com
```

实例代码：

```
# 打开文件
fo = open("runoob.txt", "r")
print ("文件名为: ", fo.name)

for line in fo.readlines():                          #依次读取每行
    line = line.strip()                             #去掉每行头尾空白
    print ("读取的数据为: %s" % (line))

# 关闭文件
fo.close()
```

输出结果：

```
文件名为:  runoob.txt
读取的数据为: 1:www.runoob.com
读取的数据为: 2:www.runoob.com
读取的数据为: 3:www.runoob.com
读取的数据为: 4:www.runoob.com
读取的数据为: 5:www.runoob.com
```

seek()函数的使用

seek() 方法用于移动文件读取指针到指定位置。

使用：fileObject.seek(offset[, whence])
offset：开始的偏移量，也就是代表需要移动偏移的字节数，如果是负数表示从倒数第几位开始。
whence：可选，默认值为0。给 offset 定义一个参数，表示要从哪个位置开始偏移：0 代表从文件开头开始算起，1 代表从当前位置开始算起，2 代表从文件末尾算起。

实例1:seek()函数的使用

假设runoob.txt文件的内容如下：

```
1:www.runoob.com
```

实例代码：

```
# 打开文件
fo = open("runoob.txt", "r+")
print ("文件名为: ", fo.name)

line = fo.readline()
print ("读取的数据为: %s" % (line))

# 重新设置文件读取指针到开头
fo.seek(0, 0)
line = fo.readline()
print ("读取的数据为: %s" % (line))


# 关闭文件
fo.close()
```

输出结果：

```
文件名为:  runoob.txt
读取的数据为: 1:www.runoob.com

读取的数据为: 1:www.runoob.com
```

write()函数的使用

write() 方法用于向文件中写入指定字符串。

在文件关闭前或缓冲区刷新前，字符串内容存储在缓冲区中，这时你在文件中是看不到写入的内容的。

实例1:write()函数的使用

假设runoob.txt文件的内容如下：

```
1:www.runoob.com
2:www.runoob.com
3:www.runoob.com
4:www.runoob.com
5:www.runoob.com
```

实例代码：

```
# 打开文件
fo = open("runoob.txt", "r+")
print ("文件名: ", fo.name)

str = "6:www.runoob.com"
# 在文件末尾写入一行
fo.seek(0, 2)
line = fo.write( str )

# 读取文件所有内容
fo.seek(0,0)
for index in range(6):
    # next()涉及到迭代器，就是会记录读到哪了，不细讲，可以自行了解。
    line = next(fo)
    print ("文件行号 %d - %s" % (index, line))

# 关闭文件
fo.close()
```

输出结果：

```
文件行号 0 - 1:www.runoob.com

文件行号 1 - 2:www.runoob.com

文件行号 2 - 3:www.runoob.com

文件行号 3 - 4:www.runoob.com

文件行号 4 - 5:www.runoob.com

文件行号 5 - 6:www.runoob.com
```

OS操作

os模块概述

os 模块提供了非常丰富的方法用来处理文件和目录。

chdir()函数的使用

os.chdir() 方法用于改变当前工作目录到指定的路径。参数中传入要切换的路径名。

实例1:chdir()切换工作目录

```
import os, sys

path = "/tmp"

# 查看当前工作目录
retval = os.getcwd()
print ("当前工作目录为 %s" % retval)

# 修改当前工作目录
os.chdir( path )

# 查看修改后的工作目录
retval = os.getcwd()

print ("目录修改成功 %s" % retval)
```

输出结果：

```
当前工作目录为 /www
目录修改成功 /tmp
```

getcwd()函数的使用

os.getcwd() 方法用于返回当前工作目录。

实例1:getcwd()获取当前工作目录

```
import os, sys

# 切换到 "/var/www/html" 目录
os.chdir("/var/www/html" )

# 打印当前目录
print ("当前工作目录 : %s" % os.getcwd())
```

输出结果：

```
当前工作目录 : /var/www/html
```

listdir()函数的使用

os.listdir() 方法用于返回指定的文件夹包含的文件或文件夹的名字的列表。这个列表以字母顺序。

实例1:listdir()得到当前目录下的文件和文件夹名字

```
import os, sys

# 打开文件
path = "/var/www/html/"
dirs = os.listdir( path )

# 输出所有文件和文件夹
for file in dirs:
    print (file)
```

输出结果：

```
test.htm
stamp
faq.htm
_vti_txt
robots.txt
itemlisting
resumelisting
writing_effective_resume.htm
advertisebusiness.htm
papers
resume
```

mkdir()函数的使用

os.mkdir() 方法用于以数字权限模式创建目录。默认的模式为 0777 (八进制)。那什么是777呢?

使用方法：os.mkdir(path[, mode])

实例1:mkdir()创建目录

```
import os, sys

# 创建的目录
path = "/tmp/home/monthly/daily/hourly"

os.mkdir( path, 0755 )

print ("目录已创建")
```

输出结果：

```
目录已创建
```

remove函数的使用

os.remove() 方法用于删除指定路径的文件。如果指定的路径是一个目录，将抛出OSError。

实例1:remove()函数删除文件或文件夹

```
import os, sys

# 列出目录
print ("目录为: %s" %os.listdir(os.getcwd()))

# 移除
os.remove("aa.txt")

# 移除后列出目录
print ("移除后 : %s" %os.listdir(os.getcwd()))
```

输出结果：

```
目录为:
[ 'a1.txt','aa.txt','resume.doc' ]
移除后 :
[ 'a1.txt','resume.doc' ]
```