nsd_1908_py01_day03

文件

文件操作的步骤：打开、读写、关闭

读取文本文件

```
(nsd1908) [root@room8pc16 day03]# cp /etc/passwd /tmp/
>>> f = open('/tmp/passwd')  # 默认以r方式打开
>>> data = f.read()   # 默认读取全部内容
>>> print(data)
>>> data = f.read()   # 继续向后读取
>>> data
''
>>> f.close()

>>> f = open('/tmp/passwd')
>>> f.read(4)  # 指定一次最多读取的字节数
'root'
>>> f.read(4)
':x:0'
>>> f.readline()  # 继续向后读取一行
':0:root:/root:/bin/bash\n'
>>> f.readlines()  # 继续向后读取，将所有的行读取到列表中
>>> f.close()

# ***重要：以下方法是最常用的遍历文件文件的方法***
>>> f = open('/tmp/passwd')
>>> for line in f:
...   print(line, end='')
>>> f.close()
```

读取非文本文件

```
>>> f = open('/tmp/aaa.jpg', 'rb')  # b表示bytes
>>> f.read(5)   # 读取5字节
b'\xff\xd8\xff\xe0\x00'    # 前面的b表示bytes
# 2进制与16进制互转：4个2进制数转换成一个16进制数
# 0000 -> 0; 0001 -> 1; 0010 -> 2; 0011 -> 3; 0100-> 4
# 1110 -> e; 1111 -> f;
>>> f.read(4096)  # 建议一次读4K或4K的倍数
>>> f.close()
```

写入文本文件

```
>>> f = open('/tmp/a.txt', 'w')
>>> f.write('hello world!\n')   # 将字符串写入文件
13   # 共写入13字节
(nsd1908) [root@room8pc16 day03]# cat /tmp/a.txt   # 文件为空
>>> f.flush()  # 立即将缓存数据同步至磁盘
(nsd1908) [root@room8pc16 day03]# cat /tmp/a.txt
hello world!
# 将字符串列表中的每个记录写到文件
>>> f.writelines(['2nd line.\n', '3rd line.\n'])
>>> f.close()
(nsd1908) [root@room8pc16 day03]# cat /tmp/a.txt
hello world!
2nd line.
3rd line.
```

以非文本形式写入文件

```
>>> f = open('/tmp/a.txt', 'wb')
# 字符串前面加b，表示这是bytes类型的数据。一个英文字符正好是一个字节。
>>> f.write(b'Hello World!\n')
13
>>> f.write(b'你好!\n')   # 错误，因为一个中文字符不是一个字节
####################
>>> b1 = '你好！\n'
>>> type(b1)
<class 'str'>
>>> b1.encode()
b'\xe4\xbd\xa0\xe5\xa5\xbd\xef\xbc\x81\n'
####################
>>> f.write(b1.encode())
10
>>> f.close()
```

字符编码

- 每个字符在计算机上存储的时候，都是使用2进制01进行表示
- 美国常用的ASCII码使用7位来表示一个字符；欧洲使用ISO－8859－1作为字符编码，它是在ASCII码基础上再扩展1位，使用8位来表示一个字符；中国常用的字符集是GB2312、GB18030、GBK
- ISO推出了unicode编码，被称作万国码。utf8编码是Unicode的一种编码方案，它采用变长的字节数表示一个字符。如1个英文字符占1个字节，一个汉字字符占3个字节。

with语句

- with打开文件，当with语句结束时，文件自动关闭

```
>>> with open('/tmp/passwd') as f:
...   f.readline()
...
'root:x:0:0:root:/root:/bin/bash\n'
>>> f.readline()   # 报错，因为文件已经关闭了
```

seek语句

- 了解即可
- 用于移动文件指针
- 它的第二个参数是数字：0->开头，1->指针当前位置，2->结尾
- 它的第一个参数是相对于第二个参数的偏移量

```
>>> f = open('/tmp/passwd', 'rb')
>>> f.seek(16, 0)  # 文件指针移动到距离开头向右16个字节处
16
>>> f.read(5)
b'/root'
>>> f.seek(-5, 2)  # 文件指针移动到距离结尾向左5个字符处
2714
>>> f.read()
b'bash\n'
>>> f.seek(0, 0)  # 文件指针移动到开头
0
>>> f.readline()
b'root:x:0:0:root:/root:/bin/bash\n'
```

拷贝文件的代码分析

```
f1 = open('/bin/ls', 'rb')
f2 = open('/tmp/list', 'wb')

data = f1.read()
f2.write(data)

f1.close()
f2.close()
```

1. 建议使用变量，而不是直接使用字面量。open的参数最好使用变量
2. 变量名应该有意义。f1、f2这样的名字不能体现出它代表什么
3. 如果文件很大，一次性的将所有内容读取出来，将会消耗太多内存。应该每次读取少量数据，多读几次

函数

函数返回值

- 函数使用return返回运行结果
- 如果没有return语句，则返回None

参数

- 函数定义时，写在()中的“变量名”叫形式参数、形参
- 调用函数时，传递给函数的具体数据，叫实际参数、实参

默认参数

- 提供了默认值的参数

```
>>> def pstar(n=30):
...   print('*' * n)
...
>>> pstar()
******************************
>>> pstar(20)
********************
```

模块

- 模块就是一个python程序文件
- 将程序文件的扩展名.py移除，剩下的文件名就是模块名

```
# vim star.py
hi = 'Hello World!'

def pstar(n=30):
    print('*' * n)

(nsd1908) [root@room8pc16 day03]# python
>>> import star
>>> star.hi
'Hello World!'
>>> star.pstar()
******************************
```

导入模块的方法

```
>>> import star  # 常用
>>> from random import choice, randint   # 常用
>>> choice('abcd')
'a'
>>> randint(1, 100)
57

>>> import sys, getpass  # 一行导入多个模块，不常用
>>> import getpass as gp  # 导入模块的同时，为它起别名，不常用
>>> gp.getpass('password: ')
```

- 导入模块时，模块文件的代码将会运行一遍，这叫加载load
- 不管导入多少次，只会加载一次

__name__特殊属性

- 每个模块都有一个特殊的变量叫__name__
- __name__变量可以有两个值
    - 如果模块文件直接运行，值为__main__

- 如果模块文件间接运行，值为模块名

```
(nsd1908) [root@room8pc16 day03]# echo 'print(__name__)' > foo.py
(nsd1908) [root@room8pc16 day03]# echo 'import foo' > bar.py
(nsd1908) [root@room8pc16 day03]# cat foo.py
print(__name__)
(nsd1908) [root@room8pc16 day03]# cat bar.py
import foo
(nsd1908) [root@room8pc16 day03]# python foo.py
__main__
(nsd1908) [root@room8pc16 day03]# python bar.py
foo
```