nsd1909-py01-day03

文件

文件操作的三个步骤：

- 打开
- 读写
- 关闭

读取文本文件

字符编码

在计算机上存储字符时，都是以2进制0/1的方式存储。将一连串的01组合表示为一个字符。

美国采用的字符编码是ASCII码；西欧使用ISO－8859－1；中国采用GBK/GB2312/GB18030。

不同国家采用的编码方案不同，导致相同的01组合在不同国家代表不同的字符。为了解决这个问题，使得某种01的组合只能代表一个字符，ISO提示了万国码Unicode，常用的Unicode解决方案是utf8。

utf8是变长的，一个英文字符，占8位，一个汉字占24位。

python中，字符有str类型和bytes类型。str就是各种字符，如英文或中文。bytes是字节。一个英文字符正好是一个字节，一个汉字是三个字节

```
>>> s1 = 'a'
>>> s2 = '中'
>>> type(s1)
<class 'str'>
>>> type(s2)
<class 'str'>
>>> s1.encode()  # 将str转为bytes类型
b'a'             # b表示bytes类型，一个英文字符正好是一个字节，就用字符本身表示
>>> s2.encode()  # 一个汉字是3个字节，转换成16进制数表示，\x表示16进制
b'\xe4\xb8\xad'
>>> bin(0xe4b8ad)  # 将16进制转成2进制，在磁盘上存'中'，就是存这些2进制数
'0b111001001011100010101101'
>>> b2 = s2.encode()
>>> b2
b'\xe4\xb8\xad'
>>> b2.decode()  # 将bytes类型转成str类型

'中'
```

```
[root@localhost day03]# cp /etc/passwd /tmp/
>>> f = open('/tmp/password')  # 默认以r方式打开，文件不存在，报错
>>> f = open('/tmp/passwd')
>>> data = f.read()   # read默认读取全部内容，返回字符串
>>> f.close()
>>> print(data)

>>> f = open('/tmp/passwd')
>>> data = f.read()   # 读取文件内容，赋值给 data
>>> data = f.read()   # 继续读取文件内容，赋值给 data
>>> f.close()
>>> data
''

>>> f = open('/tmp/passwd')
>>> f.readline()  # 读1行
'root:x:0:0:root:/root:/bin/bash\n'
>>> f.readline()  # 继续读1行
'bin:x:1:1:bin:/bin:/sbin/nologin\n'
>>> f.readlines()  # 将剩余行全部读取出来，成为字符串列表
>>> f.readline()  # 因为已全部读取完，所以再读取，已经没有数据了
''
>>> f.close()


# **********重要：最常用的读取文本文件的方法 **************
>>> 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')  # 需要以bytes方式打开
>>> f.read(10)    # 10表示，读取10字节
b'\xff\xd8\xff\xe0\x00\x10JFIF'
>>> f.close()
```

写入文本文件

```
>>> f = open('/tmp/passwd', 'w')    # 文件不存在则创建，存在则清空
>>> f.write('Hello World.\n')    # 系统将会把数据写入缓存
13    # 表示写入了13字节
[root@localhost day03]# cat /tmp/passwd     # 文件为空
# 立即同步缓存数据到磁盘。不是必须的，当数据量达到 4K时，也会写入磁盘；关闭文件时也会写入磁盘
>>> f.flush()
[root@localhost day03]# cat /tmp/passwd
Hello World.
>>> f.writelines(['2nd line.\n', '3rd line.\n'])
>>> f.close()
[root@localhost day03]# cat /tmp/passwd
Hello World.
2nd line.
3rd line.
```

写入非文本文件

```
>>> s1 = '武汉加油\n'
>>> f = open('/tmp/wh.txt', 'wb')
>>> f.write(s1)    # 报错，只能写入 bytes类型，但s1是str类型
>>> f.write(s1.encode())
13
>>> f.close()
[root@localhost day03]# cat /tmp/wh.txt
武汉加油
```

with语句

通过with语句打开文件，with语句结束时，文件自动关闭

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

移动文件指针

通过seek函数移动文件指针，它有两个参数：

- 第二个参数有三个取值：0 -> 开头，1 -> 当前位置，2 -> 结尾
- 第一个参数是相对于第二个参数的偏移量

```
[root@localhost day03]# cat /tmp/passwd
Hello World.
2nd line.
3rd line.
>>> f = open('/tmp/passwd', 'rb')
>>> f.tell()    # 总是返回文件指针相对于开头的偏移量
0
>>> f.seek(6, 0)   # 移动文件指针，从开头向后移动 6字节
6
>>> f.readline()   # 读取从件指针位置到行尾
b'World.\n'
>>> f.seek(-10, 2)  # 移动文件指针，从结尾向左移动 10字节
23
>>> f.read()    # 读取从件指针位置到结尾
b'3rd line.\n'
```

重要：读取文件方法必须熟练掌握的有：文本文件采用for循环；非文本文件采用while循环，参见文件拷贝代码

```
# 该文件非常重要，将来有很多代码都采用这种形式

src_fname = '/bin/ls'
dst_fname = '/tmp/list'
src_fobj = open(src_fname, 'rb')
dst_fobj = open(dst_fname, 'wb')

while 1:
    data = src_fobj.read(4096)  # 从源文件中一次最多读 4096字节

    # if data == b''     # 如果data为空
    # if len(data) == 0  # 如果data长度为0
    if not data:         # data为空表示False，取反为True
        break

    dst_fobj.write(data)

src_fobj.close()
dst_fobj.close()
```

函数

函数就是一堆代码集合，方便重复使用。

函数返回值

函数处理完数据后，往往有一个结果，这个结果可以使用return进行返回，如果没有return语句，那么它默认返回None。

```
>>> def add():
...     result = 10 + 20   # result在函数内定义，只能在函数内使用
...
>>> a = add()
>>> print(a)
None
>>> def add():
...     a = 10 + 5
...     b = 2 * 3
...     return 'hello world'
...
>>> r = add()
>>> r
'hello world'
```

函数参数

定义函数时，写在函数名后面括号中的内容称作形式参数。因为定义函数时，参数的值不确定，只是形式上占个位置，所以称作形式参数，简称为形参。

调用函数时，需要将具体的数据传给函数，这个具体的数据是实际使用的参数，叫做实际参数，简称为实参。

位置参数

在python中，位置参数保存在sys模块的argv列表中。

```
# vim weizhi.py
import sys

print(sys.argv)

[root@localhost day03]# python3 weizhi.py hao 123
['weizhi.py', 'hao', '123']
```

默认参数

给定了默认值的参数

```
>>> def pstar(n=30):
...     print('*' * n)
...
>>> pstar()    # 没传参，n使用默认的30
******************************
>>> pstar(50)  # n取值为50
**************************************************
```

模块

- 文件是python在物理上组织代码的形式
- 模块是python在逻辑上组织代码的形式
- 把文件名的.py扩展名去除，剩余部分就是模块名

导入模块的方法

```
# 直接导入，常用
>>> import sys
# 从模块中导入部分属性，常用
>>> from random import randint, choice
>>> randint(1, 100)
11
>>> choice('abcd')
'c'
# 导入模块时，为模块取别名，不常用
>>> import getpass as gp
>>> passwd = gp.getpass()
Password:
# 一个import语句，导入多个模块，不常用
>>> import time, sys
```

模块的加载

- 导入模块的时候，将会把模块内的代码执行一遍，这叫作加载load
- 不管导入（import）多少次，只以第一次为准

模块的特殊属性__name__

- 它是python模块自带的属性
- 双下划线只是提醒你，这是一个特殊属性
- __name__是一个变量，这个变量在不同情况下有如下的两个值：
  - 模块直接运行时，值是__main__
  - 模块被导入时，值是模块名

```
[root@localhost day03]# echo 'print(__name__)' > foo.py
[root@localhost day03]# echo 'import foo' > bar.py
[root@localhost day03]# cat foo.py
print(__name__)
[root@localhost day03]# cat bar.py
import foo
[root@localhost day03]# python3 foo.py
__main__
[root@localhost day03]# python3 bar.py
foo
```