nsd1905_py01_day03

文件对象

操作文件的步骤

- 打开文件
- 读写文件
- 关闭文件

读取文本文件

```
(nsd1905) [root@room8pc16 day03]# cp /etc/passwd /tmp/
>>> f = open('/tmp/password')  # 默认以r方式打开，文件不存在则报错
>>> f = open('/tmp/passwd')
>>> data = f.read()   # read默认读取全部数据
>>> data  # 显示data变量的内容，是一个大字符串，文件的行尾换行符使用\n表示
>>> print(data)  # print将\n转义为回车换行
>>> data = f.read()   # 这不是再读一遍，而是继续向后读取
>>> data
''
>>> f.close()  # 关闭文件

>>> f = open('/tmp/passwd')
>>> f.read(4)   # 读取4字节
'root'
>>> f.read(1)   # 读取1字节
':'
>>> f.readline()  # 从文件指针开始到该行结尾，即读一行
'x:0:0:root:/root:/bin/bash\n'
>>> f.readline()
'bin:x:1:1:bin:/bin:/sbin/nologin\n'
>>> f.readlines()   # 将文件内容读到列表中，每行是列表的一项
>>> f.close()

# 重要、常用的遍历文件的方法
>>> f = open('/tmp/passwd')
>>> for line in f:
...   print(line, end='')
>>> f.close()
```

读取非文本文件的方法

```
>>> f = open('/bin/ls')
>>> f.read(10)  # python试图将读取的10个字节转换成字符，但是无法转换，报错
>>> f.close()

>>> f = open('/bin/ls', 'rb')   # b表示bytes类型。
# 读取10个字节，如果一个字节正好可以表示为一个英文字符，就以字符(str)显示，否则，将1个字节转换成2个16进制数表示。
>>> f.read(10)  # 建议读4096的倍数那么多数据
b'\x7fELF\x02\x01\x01\x00\x00\x00'
>>> f.close()
```

写入文本文件

```
>>> f = open('/tmp/passwd', 'w')  # 以w方式打开文件，将清空或创建文件
(nsd1905) [root@room8pc16 day03]# cat /tmp/passwd   # 空文件
>>> f.write('hello world!\n')
13  # 表示写入13字节
(nsd1905) [root@room8pc16 day03]# cat /tmp/passwd   # 仍然无内容
>>> f.flush()   # 立即将数据同步至磁盘
(nsd1905) [root@room8pc16 day03]# cat /tmp/passwd
hello world!
>>> f.writelines(['new line.', 'ni hao', 'how are you?'])
>>> f.close()
# 多出一行，因为写入时没有使用\n
(nsd1905) [root@room8pc16 day03]# cat /tmp/passwd
```

写入非文本文件(以**bytes**类型写入)

```
>>> s1 = 'hello 中国'
>>> type(s1)   # 无论是英文还是中文字符，都是字符str
<class 'str'>
>>> s1.encode()  # 将str类型转为bytes类型
b'hello \xe4\xb8\xad\xe5\x9b\xbd'
>>> data = s1.encode()
>>> type(data)
<class 'bytes'>
>>> data
b'hello \xe4\xb8\xad\xe5\x9b\xbd'
>>> data.decode()   # 将bytes类型转成str类型
'hello 中国'
>>> f = open('/tmp/mytest.txt', 'wb')
>>> f.write(data)
12
>>> f.close()
```

with语句

通过with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。
```

seek移动文件指针

```
# seek(偏移量, 位置)  位置：0表示开头，1表示当前位置，2表示结尾
>>> f = open('/tmp/passwd', 'rb')
>>> f.seek(16, 0)   # 从开头向右移动16字节
16
>>> f.read(5)
b'/root'
>>> f.seek(-16, 2)   # 从结尾向左移动16字节
2740   # 2740是从开头到指针位置的偏移量
>>> f.read()
b'jerry:/bin/bash\n'
>>> f.close()
```

函数

- 函数一般用于实现某一功能
- 它实现了代码的重用
- 函数定义时，函数内的代码块不会执行
- 执行函数内的代码，需要通过一对小括号进行调用

```
def 函数名(参数):
    代码块
```

返回值

- 返回值就是函数处理数据的结果
- 使用关键字return来返回结果
- 如果没有return语句，默认返回None
- 函数的返回值，可以返回任意的数据，但是应该合理

参数

- 参数是可选的
- 函数需要处理的数据应该通过参数进行传递

```
>>> def show(data):
...   print('数据:', data)
...
>>> with open('/tmp/passwd') as f:
...   for line in f:
...     show(line)

>>> a = input('data: ')
data: hello world
>>> show(a)
数据: hello world
```

位置参数

- python将位置参数保存到了sys模块的argv列表
- 与shell的$1 / $2等表示一样的含义
- 位置参数的数据类型都是字符串

默认参数

- 为参数添加了默认值

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

模块

- 一个以.py结尾的程序文件就是一个模块
- 文件是python物理上组织代码的形式
- 模块是逻辑上组织代码的形式
- 模块名也是一个名字，可用字符的要求与变量名一样

```
# vim star.py
'''打印星号

这是一个演示用的模块，只包含了一个全局变量和一个函数
'''

hi = 'Hello World'

def pstar(n=30):
    '缺省打印30个星号'
    print('*' * n)

>>> import star
```

```
>>> star.hi
'Hello World'
>>> star.pstar()
******************************
>>> help(star)  # 查看模块的帮助信息
```

导入模块的方法

```
# 直接导入，常用
>>> import sys
>>> sys.argv
['']

# 仅导入模块中的某些方法，常用
>>> from random import choice, randint
>>> choice('abcd')
'a'
>>> randint(1, 100)
54

# 一行导入多个模块，不常用
>>> import time, os

# 导入模块时，为模块定义别名，不常用1
>>> import getpass as gp
>>> a = gp.getpass()
Password:
>>> a
'abc'
```

模块的导入和加载

- import是导入模块
- 第一次导入模块时，模块中的代码会执行一遍，这个叫加载，即load
- 无论导入多少次，只会加载一次

模块导入的特性

- 每个模块都有一个名为__name__的变量
- __name__它的值，可能是__main__，也可能是模块名
    - 当程序文件作为脚本，直接运行时，值为__main__
    - 当程序文件作为模块，被导入时，值为模块名

```
(nsd1905) [root@room8pc16 day03]# cat foo.py
print(__name__)
(nsd1905) [root@room8pc16 day03]# cat bar.py
import foo
(nsd1905) [root@room8pc16 day03]# python foo.py
__main__
(nsd1905) [root@room8pc16 day03]# python bar.py
foo
```