nsd1904_py01_day03

文件

操作的三个步骤

- 打开文件
- 读写文件
- 关闭文件

读取文本文件的方法

```
(nsd1904) [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()   # 默认把文件所有内容读取进来
>>> f.close()   # 关闭文件
>>> print(data)

>>> f = open('/tmp/passwd')
>>> f.read(4)   # 指定读取的字节数
'root'
>>> f.read(3)
':x:'
>>> f.readline()   # 读取一行（找到\n算一行）
'0:0:root:/root:/bin/bash\n'
>>> f.readline()
'bin:x:1:1:bin:/bin:/sbin/nologin\n'
>>> lines = f.readlines()  # 读取全部行，存入列表，每一行是一个列表项
>>> lines
>>> f.close()

# 文本文件常用处理方式。重要
>>> f = open('/tmp/passwd')
>>> for line in f:
...   print(line, end='')
>>> f.close()
```

读取非文本文件

```
>>> f = open('/bin/ls', 'rb')   # b表示bytes类型
>>> f.read(10)  # 读10字节，如果读取的内容，一个字节刚好可以显示成一个字符，就显示字符，否则将显示该字节的16进制数，\x表示16进制
b'\x7fELF\x02\x01\x01\x00\x00\x00'  # b表示bytes
>>> data = f.read(4096)   # 建议每次读4096的倍数
>>> f.close()
```

写入文件

```
>>> f = open('/tmp/passwd', 'w')  # 清空或创建文件
# 系统将数据写入缓存，数据量较大时，或关闭文件时，内容才会同步至磁盘
>>> f.write('hello world!\n')
13     # 表示写入了13字节
>>> f.flush()   # 立即写入磁盘
>>> f.writelines(['2nd line.\n', '3rd line.\n'])  # 写入字符串列表中数据
>>> f.close()
```

with语句

通过with打开文件，with语句结束后，文件自动关闭

```
>>> with open('/tmp/passwd') as f:
...   f.readline()
...
'hello world!\n'
>>> f.readline()   # 文件已关闭，无法读取
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file.
```

seek移动文件指针

```
>>> f = open('/tmp/passwd', 'rb')
>>> f.seek(6, 0)   # 从开头向后偏移6字节
6
>>> f.read(5)   # 读取5字节数据
b'world'
>>> f.seek(2, 1)   # 从当前位置向后移2字节
>>> f.read(3)
b'2nd'
>>> f.seek(-6, 2)   # 移动指针到结尾前6字节处
27
>>> f.read()
b'line.\n'
>>> f.close()
```

函数

```
def 函数名(参数...):
    函数体内的代码
```

- 函数定义时，里面的代码不会执行
- 函数调用，务必通过()调用
- 调用时，将会执行函数内的代码

```
>>> def pstar():
...   print('*' * 50)
...
>>> pstar   # 交互解释器将显示函数在内存中的位置
<function pstar at 0x7faf48e6e2f0>
>>> pstar()   # 调用函数
**************************************************
```

函数的返回值

函数的返回值就是函数运行后的结果，通过return返回，如果没有return则返回None。

```
>>> def add():
...   a = 10 + 5
...
>>> x = add()
>>> print(x)
None

>>> def add():
...   a = 10 + 5
...   return 'Hello World'
...
>>> x = add()
>>> print(x)
Hello World

>>> def add1():
...   print(10 + 5)
...
>>> x = add1()
15
>>> x * 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for *: 'NoneType' and 'int'
>>> def add2():
...   return 10 + 5
...
>>> y = add2()
>>> y * 2
30
```

函数参数

- 定义函数时，函数名字后面括号中填写的是参数。
- 函数处理的数据，应该通过参数传递。

位置参数

- 在python中，命令行上的位置参数保存到了sys模块的argv列表中

默认参数

给定了默认值的参数，就是默认参数

```
>>> def pstar(n=30):
...   print('*' * n)
...
>>> pstar()
******************************
>>> pstar(50)
**************************************************
>>> pstar()
******************************
```

模块

- 一个以.py结尾的python程序文件就是一个模块
- 把文件名的.py移除，剩余的文件名部分就是模块名
- 文件是代码的物理组织形式
- 模块是代码的逻辑组织形式

```
# 导入模块的方法
# 1. 每行导入一个模块，常用
>>> import time
>>> import getpass
# 2. 仅导入模块中的某些功能，常用
>>> from random import choice, randint
>>> randint(1, 10)
1
>>> choice('abcd')
'd'
# 3. 一行导入多个模块，不建议
>>> import sys, os, abc
# 4. 导入模块时，为其起别名，不常用
>>> import getpass as gp
>>> a = gp.getpass()
Password:
```

- import是导入模块
- 导入模块时，将会导致模块文件的代码执行，称作加载load
- 不管import导入模块多少遍，仅load一次

自定义模块

```
# vim star.py
"""打印星号

该模块是一个演示用的模块。它包含一个全局变量和一个函数。
"""
hi = 'Hello World!'
```

```
def pstar(n=30):
    "用于打印一行星号"
    print('*' * n)

pstar()

# python
>>> import star
>>> star.hi
>>> star.pstar()
>>> star.pstar(50)
>>> help(star)
```

模块特性__name__

- 每个模块都有一个特殊的变量叫__name__
- __name__值在不同的情况下不同
  - 如果模块文件直接运行，值为__main__
  - 如果模块因为import间接运行，值为模块名

```
(nsd1904) [root@room8pc16 day03]# cat aaa.py
print(__name__)
(nsd1904) [root@room8pc16 day03]# cat bbb.py
import aaa
(nsd1904) [root@room8pc16 day03]# python aaa.py
__main__
(nsd1904) [root@room8pc16 day03]# python bbb.py
aaa
```

总结

文件

处理文本文件：for循环

处理非文本文件：while循环

函数

定义：def

返回值：return

参数：函数需要处理的数据，用参数传递。位置参数、默认参数。

模块

导入：import

加载：load，模块中的代码执行一次

导入方法：import xxx; from xxx import yyy

模块的特殊属性：__name__