一、文件对象

文件操作的三个步骤：打开文件、读写、关闭文件

|  |  |
| --- | --- |
| **open模式** | **操作** |
| r | 以读方式打开(文件不存在则报错) |
| w | 以写方式打开(文件存在则清空,不存在则创建) |
| a | 以追加模式打开(必要时创建新文件) |
| r+ | 以读写模式打开(参见r) |
| w+ | 以读写模式打开(参见w) |
| a+ | 以读写模式打开(参见a) |
| b | 以二进制模式打开 |

**1、读文件**

>>> fobj = open('/tmp/passwd')

#默认以r方式打开

>>> data = fobj.read()

#读取文件所有内容，返回str

>>> data = fobj.read(4096)

#建议读4096字节，现在硬盘的扇区大小一般为4K

#每次读取文件，指针向后移动。

>>> fobj.readline()

#读一行，返回str

>>> fobj.readlines()

#将所有行读到一个列表中，返回list（每行一个str值）

>>> for line in fobj:

... print(line, end='')

#for循环历遍

>>> fobj.read('/bin/ls', 'rb')

#二进制文件包括：命令程序、图片、视频、声音

>>> fobj.close()

1. **写文件**

>>> fobj = open('/tmp/passwd', 'w')

#注意w方式打开，如果文件存在则清空；文件不存在则创建

>>> fobj.write('hello world!\n')

#数据没有立即写入磁盘，而是在缓冲区积累到一定量才同步到磁盘

#fobj.write()只能写入字符串

>>> fobj.flush()

#立即同步到磁盘

>>> fobj.writelines(['2nd line.\n', '3rd line.\n'])

#fobj.writelines可以写入列表

>>> fobj.close()

#关闭文件，并立即同步到磁盘

1. **操作文件**

1）with语句：自动关闭文件

>>> with open('/tmp/passwd') as fobj:

... fobj.readline()

>>> fobj.read()

#报错，因为文件已关闭

1. 文件内移动（非重点）

#seek的第一个参数是偏移量

#seek的第二个参数是相对位置，0表示开头，1表示当前位置，2表示结尾

>>> fobj = open('/tmp/passwd', 'rb')

#若想支持左移、从当前位置移动，必须使用b方式打开

>>> fobj.seek(6, 0)

#从开头向右移6字节

>>> fobj.seek(2, 1)

#从当前位置向右移2字节

>>> fobj.seek(-6, 2)

#从结尾向左移6字节

>>> fobj.read()

>>> fobj.tell()

#返回当前文件指针的位置

3）案例：模拟cp操作

src\_fname = '/bin/ls'

dst\_fname = '/root/ls'

with open(src\_fname, 'rb') as src\_fobj:

with open(dst\_fname, 'wb') as dst\_fobj:

while True:

data = src\_fobj.read(4096) #每次读4096字节

# if len(data) == 0:

# if data == b'':

if not data:

break

dst\_fobj.write(data)

二、函数

**1、函数定义、调用、返回值**

1）案例：使用函数生成斐波那契数列

def gen\_fib(len):

alist = [0, 1]

for i in range(len-2):

alist.append(alist[-1] + alist[-2])

return alist

#没有return的话，默认返回None

print(gen\_fib(10))

#不加()为引用函数，加()为调用函数

b= gen\_fib(20)

with open('/tmp/fib.txt', 'w') as fobj:

fobj.write(str(b))

1. **位置参数**
2. 使用sys.argv列表

[root@room9pc01 ~]# vim /root/argv.py

import sys

print(sys.argv) #sys.argv是一个列表

for i in range(len(sys.argv)):

print(sys.argv[i])

[root@room9pc01 ~]# python3 /root/argv.py hello 123

['/root/argv.py', 'hello', '123']

/root/argv.py

hello

123

#注意：'123'是字符串类型，sys.argv只读入字符串类型

2）案例：使用位置变量，实现拷贝文件功能

import sys

def copy(src\_fn, dst\_fn):

with open(src\_fn, 'r') as src\_fobj:

with open(dst\_fn, 'w') as dst\_fobj:

while True:

data = src\_fobj.read(4096)

if not data:

break

dst\_fobj.write(data)

copy(sys.argv[1], sys.argv[2])

**3、默认参数**

1）案例：定义'\*'的个数

def pstar(num = 30):

print('\*' \* num)

pstar()

pstar(50)

三、模块

一个以.py作为扩展名的python程序文件就是一个模块。

注意：自定义模块名不要与系统模块名重名

1. **模块创建、模块导入、模块加载**

[root@room9pc01 ~]# cat star.py #创建模块

hi = 'hello world'

def pstar(n = 30):

print('\*' \* n)

[root@room9pc01 ~]# python3

>>> import star

>>> print(star.hi) #使用模块变量

hello world

>>> star.pstar(50) #使用模块函数

#一个模块只被加载一次,无论它被导入多少次

**2、模块导入的特性**

模块具有一个\_\_name\_\_的特殊属性；

当模块文件直接执行，\_\_name\_\_ = '\_\_main\_\_'

当模块被另一文件导入时，\_\_name\_\_ = '模块名'

1）\_\_name\_\_变量特性

[root@room9pc01 ~]# cat foo.py

print(\_\_name\_\_)

[root@room9pc01 ~]# cat bar.py

import foo

[root@room9pc01 ~]# python3 foo.py

\_\_main\_\_

[root@room9pc01 ~]# python3 bar.py

foo

1. 如何让模块调用时，不执行模块的输出命令

[root@room9pc01 ~]# cat star.py #创建模块

hi = 'hello world'

def pstar(n = 30):

print('\*' \* n)

if \_\_name\_\_ == '\_\_main\_\_':

pstar(50)

#Pycharm软件的快捷键：main[tab]

3）案例：生成随机密码

import random

import string

all\_chs = string.ascii\_letters + string.digits

#大小写字母+数字

def gen\_pass(n=8):

result = ''

for i in range(n):

ch = random.choice(all\_chs)

result += ch

return result

if \_\_name\_\_ == '\_\_main\_\_':

a = gen\_pass()

print(a)

Tips：按住Ctrl键，点击"string"调用模块名可以打开string.py