# 一、列表字典元组集合

列表（list），元组（tutlp），集合（set），字典（dict）

[v],(v),{v},{k:v}

## 列表

（1）**使用filter函数**

filter（lambda x:x>=0， data）函数

（2）**列表解析**

[x for x in data if x >= 0] 更快

（3）**字典解析**

{x:randint(20,100) for x in range(3)}

{1:20,2:40,3:99}

（4）**集合**

{x for x in data if x >= 0}

（5）**元组：速度快,空间小**

元组起名字

status = （“1”,”vv”）

Id, name = range(2) 从新赋值

status[Id]

status[name]

from collections import nametuple

student = nametuple(“student”,[‘1’,’2’])

s = student(“v”,”v1”)

s.v,s.v1

## 2、统计序列中元素的频度

c = dict.fromkeys(data, 0)

for x in data:

c[x] +=1

c统计结果

==========

from collections import Counter

c2 = Counter(data) //会生成一个构造器

c2.most\_common(n) //前N名最高的

词频统计

txt = open(“xx.txt”).read()

re.split(“\w+”, txt) # 分割

c3 = counter(te)

## 3、字典项进行排序

sorted([]) 列表排序

按v排序

元组比较，先比较第0个位置，依次类排

b = zip（d.values(), d.keys()） #将字典的value放到第一位，压缩

sorted(b)

方法二

a字典，a.items转换为元组

sorted(a.items(),key=lambda n:n[0])

sorted(a,key=lambda n:n[1])

## 4、快速找个字典中的公共建

1. 多次迭代
2. map映射函数

## 5、如何让字典保持有序

{a:(1,2), b(3,4), c(e,d)} #字典是无需的

from collections import OrdereDict

d = OrdereDict() # 有序字典

## 6、随机数

from random import randint, sample

sample(“abcde”,4) #在abcde中随机获取几个数

## 7、用户历史记录

只显示固定条数的记录（容量为N的队列）

from collections import deque 双端队列

q = deque([].5) # 生成一个容量为5的队列

q.append(1) # 入队

import pickle # 将python对象保存在文本中

pickle.dump(obj, file)

pickle.load(file)

## 8、如何实现可迭代对象和迭代器对象

数据过大，用时访问

可迭代对象，（for x in a; a是可迭代对象）

迭代器对象，通过iter()将可迭代对象转换为迭代器对象

列表，字符串，可迭代对象

因为完成了\_\_iter\_\_方法

## 9、如何使用生成器函数实现可迭代对象

# 二、字符串

## 1、如何拆分含有多种分隔符的字符串

单一分隔符

s.split() # 每次处理1次字符，处理多次map函数

使用正则

## 2、如何判断是否以a开头或以B结尾

s.endswith((“.sh”,”.o”))

oct() #8进制数

s.startswith()

## 3、如何调整字符串中文件格式

## 4、如何将小字符串拼接成大字符串

将字符串存放，for循环，会造成资源浪费

使用str.join([])合并，节省资源

## 5、如何对字符串左右对齐

## 6、去掉不需要的字符

s.strip() # 默认删除空白字符

s.lstrip()

## 7、如何读取文本

python2，写入文件前对文本进行unicode编码，读入文件后对二进制字符串解码

python3 open函数指定”t”的文件模式，endcoding指定编码格式。

unicode，python中使用同一的Unicode（python3是utf-8）

encode():对uncode格式进行编码，编码成string格式\x

decode（）：对string格式转换为uncode或指定编码格式

python中“sss”就是Unicode或utf-8 ,b”sss”是字节

## 8、如何操作二进制文件

## 9、如何设置文件缓冲

操作系统使用缓冲区写入文件，每一块缓冲区有4096字节大小，超过这个大小，就写入磁盘

open(“demo.txt”,”w”,buffering=1024) buffering缓冲区大小

## 10、如何将文件映射到内存

## 11、读写csv数据

from csv

a = open(“csv”,”rb”)

csv.reader(a)

csv.next() #csv是迭代对象

csv.write()

## 12、读写json数据

# 三、类

## 1、如何派生内置不可变类型，改变内置元组的实例化行为

7-1

定义类inttuple继承内置tuple，并实现\_\_new\_\_,修改实例化行为

class IntTuple(tuple):

def \_\_new\_\_(cls, info):

# cls传入的是IntTuple类

# 修改info

retunr super(IntTule，cls）.\_\_new\_\_(cls,info)

## 2、如何为创建大量实例节省内存

方案：定义类的的\_\_slots\_\_属性，用它来声明实例属性名字列表

对象的.\_\_dict\_\_动态绑定属性

p1.\_\_dict\_\_[‘x’] = 1,del p1.\_\_dict\_\_[‘x’]

## 3、如何让对象支持上下文管理

7-2.3 telnet服务器

典型的with open（）语句

定义 \_\_enter\_\_,\_\_exit\_\_这里方法，分别会在开始，结束的时候调用

class Telnet(object):

def \_\_enter\_\_(self):

return self

def \_\_exit\_\_(self,exc\_type,exc\_val,exc\_tb):

# 异常类型，异常值，异常

return true# 可以阻止抛出异常

with Telnet() as clent:

clent.start()

## 4、如何创界可管理的对象属性

直接访问属性，没有做属性类型判断

使用方法设置属性，=》可对属性值做判断，防止之后出错，数据安全

使用内置@property()

R = property(访问方法， 设置方法）

对应print(R), R = 1

## 5、如何让类支出比较方法

7-5

实现\_\_lt\_\_,\_\_le\_\_,\_\_gt\_\_,\_\_ge\_\_,\_\_eg\_\_,\_\_ne\_\_等方法

## 6、如何使用描述符对实例属性做检查

实现\_\_get\_\_/\_\_set\_\_/\_\_delete\_\_

在\_\_set\_\_内使用isinstance

## 7、如何在环状数据结构中管理内存

7-7

## 8、如何通过实例方法名字的字符串调用方法

getattr() #内置方法