Yuan先生

随笔 - 1 文章 - 141 评论 - 143

Py西游攻关之基础数据类型

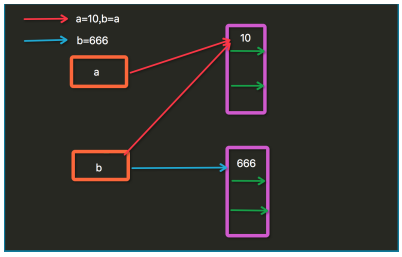
数据类型

计算机顾名思义就是可以做数学计算的机器，因此，计算机程序理所当然地可以处理各种数值。但是，计算机能处理的远不止数值，还可以处理文本、图形、音频、视频、网页等各种各样的数据，不同的数据，需要定义不同的数据类型。在Python中，能够直接处理的数据类型有以下几种

一 Number(数字)

1.1 数字类型的创建

```
a=10
b=a
b=666

print(a)#10
print(b)#666
```



注意这里与C的不同：

```
#include <stdio.h>
void main(void)
{

    int a = 1;
    int b = a;
    printf ("a:adr:%p,val:%d,b:adr:%p,val:%d\n",&a,a,&b,b);
    a = 3;
    printf ("a:adr:%p,val:%d,b:adr:%p,val:%d\n",&a,a,&b,b);

}

//打印结果：
topeet@ubuntu:~$ gcc test.c
topeet@ubuntu:~$ ./a.out
a:adr:0x7fff343a069c,val:1
b:adr:0x7fff343a0698,val:1
a:adr:0x7fff343a069c,val:3
b:adr:0x7fff343a0698,val:1
```

1.2 Number 类型转换

```
var1=3.14
var2=5
var3=int(var1)
var4=float(var2)

print(var3,var4)
```

PY内置数学函数

二 字符串类型（string）

字符串是以单引号'或双引号"括起来的任意文本，比如'abc'，"123"等等。

请注意，''或""本身只是一种表示方式，不是字符串的一部分，因此，字符串'abc'只有a，b，c这3个字符。如果'本身也是一个字符，那就可以用""括起来，比如"I'm OK"包含的字符是I，'，m，空格，O，K这6个字符。

2.1 创建字符串：

```
var1 = 'Hello World!'
var2 = "Python RAlvin"
```

对应操作：

```
# 1   * 重复输出字符串
print('hello'*2)

# 2 [] ,[:] 通过索引获取字符串中字符,这里和列表的切片操作是相同的,具体内容见列表
print('helloworld'[2:])

# 3 in  成员运算符 - 如果字符串中包含给定的字符返回 True
print('el' in 'hello')

# 4 %   格式字符串
print('alex is a good teacher')
print('%s is a good teacher'%'alex')


# 5 +   字符串拼接
a='123'
b='abc'
c='789'
d1=a+b+c
print(d1)
# +效率低,该用join
d2=''.join([a,b,c])
print(d2)
```

python的内置方法

View Code

三 字节类型(bytes)

```
# a=bytes('hello','utf8')
# a=bytes('中国','utf8')


a=bytes('中国','utf8')
b=bytes('hello','gbk')
#
print(a)        #b'\xe4\xb8\xad\xe5\x9b\xbd'
print(ord('h')) #其十进制 unicode 值为: 104
print(ord('中'))#其十进制 unicode 值为:20013

#  h   e  l   l   o
# 104 101 108 108 111   编码后结果:与ASCII表对应


#     中               国
#   \xd6\xd0         \xb9\xfa       gbk编码后的字节结果
#\xe4 \xb8 \xad   \xe5 \x9b \xbd    utf8编码后的字节结果
# 228 184 173      229 155  189        a[:]切片取


c=a.decode('utf8')
```

```
d=b.decode('gbk')
#b=a.decode('gbk') :很明显报错

print(c) #中国
print(d) #hello
```

注意：对于 ASCII 字符串，因为无论哪种编码对应的结果都是一样的，所以可以直接使用 b'xxxx' 赋值创建 bytes 实例，但对于非 ASCII 编码的字符则不能通过这种方式创建 bytes 实例，需要指明编码方式。

```
b1=b'123'
print(type(b1))
# b2=b'中国' #报错
# 所以得这样:
b2=bytes('中国','utf8')
print(b2)#b'\xe4\xb8\xad\xe5\x9b\xbd'
```

四 布尔值

一个布尔值只有True、False两种值，要么是True，要么是False，在Python中，可以直接用True、False表示布尔值（请注意大小写）

```
print(True)
print(4>2)
print(bool([3,4]))
print(True+1)
```

与或非操作：

```
bool(1 and 0)
bool(1 and 1)
bool(1 or 0)
bool(not 0)
```

布尔值经常用在条件判断中:

```
age=18
if age>18:#bool(age>18)
    print('old')
else:
    print('young')
```

五 List（列表）

OK,现在我们知道了字符串和整型两个数据类型了，那需求来了，我想把某个班所有的名字存起来，怎么办?

有同学说，不是学变量存储了吗，我就用变量存储呗，呵呵，不嫌累吗，同学，如班里有一百个人，你就得创建一百个变量啊，消耗大，效率低。

又有同学说，我用个大字符串不可以吗，没问题，你的确存起来了，但是，你对这个数据的操作（增删改查）将变得非常艰难，不是吗，我想知道张三的位置，你怎么办?

在这种需求下，编程语言有了一个重要的数据类型 - - - - 列表（list）

什么是列表：

列表（list）是Python以及其他语言中最常用到的数据结构之一。Python使用使用中括号 [ ] 来解析列表。列表是可变的（mutable）——可以改变列表的内容。

对应操作：

1 查 ([])

```
names_class2=['张三','李四','王五','赵六']

# print(names_class2[2])
# print(names_class2[0:3])
```

```
# print(names_class2[0:7])
# print(names_class2[-1])
# print(names_class2[2:3])
# print(names_class2[0:3:1])
# print(names_class2[3:0:-1])
# print(names_class2[:])
```

2 增（append，insert）

insert 方法用于将对象插入到列表中，而append方法则用于在列表末尾追加新的对象

```
names_class2.append('alex')
names_class2.insert(2,'alvin')
print(names_class2)
```

3 改（重新赋值）

```
names_class2=['张三','李四','王五','赵六']

names_class2[3]='赵七'
names_class2[0:2]=['wusir','alvin']
print(names_class2)
```

4 删（remove，del，pop）

```
names_class2.remove('alex')
del names_class2[0]
del names_class2
names_class2.pop()#注意,pop是有一个返回值的
```

5 其他操作

5.1 count

count 方法统计某个元素在列表中出现的次数：

```
>>> ['to', 'be', 'or', 'not', 'to', 'be'].count('to')
2
>>> x = [[1,2], 1, 1, [2, 1, [1, 2]]]
>>> x.count(1)
2
>>> x.count([1,2])
1
```

5.2 extend

extend 方法可以在列表的末尾一次性追加另一个序列中的多个值。

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6]
```

extend 方法修改了被扩展的列表，而原始的连接操作（+）则不然，它会返回一个全新的列表。

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6]
>>>
>>> a + b
[1, 2, 3, 4, 5, 6, 4, 5, 6]
>>> a
[1, 2, 3, 4, 5, 6]
```

5.3 index

index 方法用于从列表中找出某个值第一个匹配项的索引位置：

```
names_class2.index('李四')
```

5.4 reverse

reverse 方法将列表中的元素反向存放。

```
names_class2.reverse()
print(names_class2)
```

5.5 sort

sort 方法用于在原位置对列表进行排序。

```
x = [4, 6, 2, 1, 7, 9]
x.sort()#x.sort(reverse=True)
```

5.6 深浅拷贝

现在，大家先不要理会什么是深浅拷贝，听我说，对于一个列表，我想复制一份怎么办呢?

肯定会有同学说，重新赋值呗:

```
names_class1=['张三','李四','王五','赵六']
names_class1_copy=['张三','李四','王五','赵六']
```

这是两块独立的内存空间

这也没问题，还是那句话，如果列表内容做够大，你真的可以要每一个元素都重新写一遍吗？当然不啦，所以列表里为我们内置了copy方法:

```
names_class1=['张三','李四','王五','赵六',[1,2,3]]
names_class1_copy=names_class1.copy()

names_class1[0]='zhangsan'
print(names_class1)
print(names_class1_copy)

############
names_class1[4][2]=5
print(names_class1)
print(names_class1_copy)

#问题来了,为什么names_class1_copy,从这一点我们可以断定,这两个变量并不是完全独立的,那他们的关系是什么呢?为
```

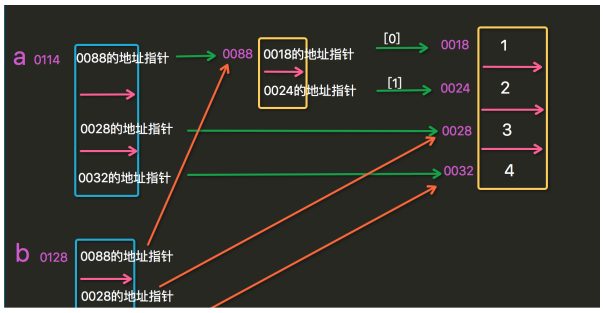
这里就涉及到我们要讲的深浅拷贝了:

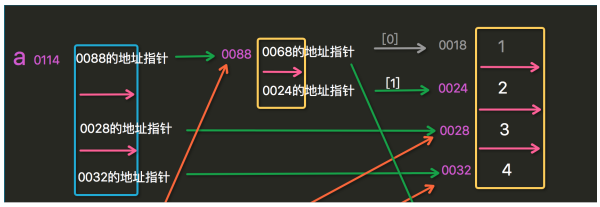
```
#不可变数据类型:数字,字符串,元组          可变类型:列表,字典

# l=[2,2,3]
# print(id(l))
# l[0]=5
# print(id(l))   # 当你对可变类型进行修改时,比如这个列表对象l,它的内存地址不会变化,注意是这个列表对象l,不
#                # this is the most important
#
# s='alex'
# print(id(s))   #像字符串,列表,数字这些不可变数据类型,,是不能修改的,比如我想要一个'Alex'的字符串,只能重
#
# s[0]='e'       #报错
# print(id(s))

#重点:浅拷贝
a=[[1,2],3,4]
b=a[:]#b=a.copy()

print(a,b)
print(id(a),id(b))
print('**************')
print('a[0]:',id(a[0]),'b[0]:',id(b[0]))
print('a[0][0]:',id(a[0][0]),'b[0][0]:',id(b[0][0]))
print('a[0][1]:',id(a[0][1]),'b[0][1]:',id(b[0][1]))
print('a[1]:',id(a[1]),'b[1]:',id(b[1]))
print('a[2]:',id(a[2]),'b[2]:',id(b[2]))
```

```
print('_____________________________________')
b[0][0]=8

print(a,b)
print(id(a),id(b))
print('*************')
print('a[0]:',id(a[0]),'b[0]:',id(b[0]))
print('a[0][0]:',id(a[0][0]),'b[0][0]:',id(b[0][0]))
print('a[0][1]:',id(a[0][1]),'b[0][1]:',id(b[0][1]))
print('a[1]:',id(a[1]),'b[1]:',id(b[1]))
print('a[2]:',id(a[2]),'b[2]:',id(b[2]))<br><br><br>#outcome
```

```
# [[1, 2], 3, 4] [[1, 2], 3, 4]
# 4331943624 4331943752
# *************
# a[0]: 4331611144 b[0]: 4331611144
# a[0][0]: 4297375104 b[0][0]: 4297375104
# a[0][1]: 4297375136 b[0][1]: 4297375136
# a[1]: 4297375168 b[1]: 4297375168
# a[2]: 4297375200 b[2]: 4297375200
# _____________________________________
# [[8, 2], 3, 4] [[8, 2], 3, 4]
# 4331943624 4331943752
# *************
# a[0]: 4331611144 b[0]: 4331611144
# a[0][0]: 4297375328 b[0][0]: 4297375328
# a[0][1]: 4297375136 b[0][1]: 4297375136
# a[1]: 4297375168 b[1]: 4297375168
# a[2]: 4297375200 b[2]: 4297375200
```

那么怎么解释这样的一个结果呢?





再不懂，俺就没办法啦...

列表补充:

```
b,*c=[1,2,3,4,5]
```

六 tuple (元组)

元组被称为只读列表，即数据可以被查询，但不能被修改，所以，列表的切片操作同样适用于元组。

元组写在小括号(())里，元素之间用逗号隔开。

虽然tuple的元素不可改变，但它可以包含可变的对象，比如list列表。

构造包含 0 个或 1 个元素的元组比较特殊，所以有一些额外的语法规则：

```
tup1 = ()    # 空元组
tup2 = (20,) # 一个元素，需要在元素后添加逗号
```

作用：

1 对于一些数据我们不想被修改，可以使用元组；

2 另外，元组的意义还在于，元组可以在映射（和集合的成员）中当作键使用——而列表则不行；元组作为很多内建函数和方法的返回值存在。

字典

⊞ 购物车实例

七 Dictionary（字典）

字典是python中唯一的映射类型，采用键值对（key-value）的形式存储数据。python对key进行哈希函数运算，根据计算的结果决定value的存储地址，所以字典是无序存储的，且key必须是可哈希的。可哈希表示key必须是不可变类型，如：数字、字符串、元组。

字典(dictionary)是除列表意外python之中最灵活的内置数据结构类型。列表是有序的对象结合，字典是无序的对象集合。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。

创建字典：

```
dic1={'name':'alex','age':36,'sex':'male'}
dic2=dict((('name','alex'),))
print(dic1)
print(dic2)
```

对应操作：

1 增

```
dic3={}

dic3['name']='alex'
dic3['age']=18
print(dic3)#{'name': 'alex', 'age': 18}

a=dic3.setdefault('name','yuan')
b=dic3.setdefault('ages',22)
print(a,b)
print(dic3)
```

2 查

```
dic3={'name': 'alex', 'age': 18}

# print(dic3['name'])
# print(dic3['names'])
#
# print(dic3.get('age',False))
# print(dic3.get('ages',False))

print(dic3.items())
print(dic3.keys())
print(dic3.values())

print('name' in dic3)# py2:  dic3.has_key('name')
print(list(dic3.values()))
```

3 改

```
dic3={'name': 'alex', 'age': 18}

dic3['name']='alvin'
```

```
dic4={'sex':'male','hobby':'girl','age':36}
dic3.update(dic4)
print(dic3)
```

4 删

```
dic4={'name': 'alex', 'age': 18,'class':1}


# dic4.clear()
# print(dic4)
del dic4['name']
print(dic4)

a=dic4.popitem()
print(a,dic4)

# print(dic4.pop('age'))
# print(dic4)

# del dic4
# print(dic4)
```

5 其他操作以及涉及到的方法

5.1 dict.fromkeys

```
d1=dict.fromkeys(['host1','host2','host3'],'Mac')
print(d1)

d1['host1']='xiaomi'
print(d1)
#######
d2=dict.fromkeys(['host1','host2','host3'],['Mac','huawei'])
print(d2)
d2['host1'][0]='xiaomi'
print(d2)
```

5.2 d.copy() 对字典 d 进行浅复制，返回一个和d有相同键值对的新字典

5.3 字典的嵌套

View Code

5.4 sorted(dict) : 返回一个有序的包含字典所有key的列表

```
dic={5:'555',2:'222',4:'444'}
print(sorted(dic))
```

5.5 字典的遍历

```
dic5={'name': 'alex', 'age': 18}

for i in dic5:
    print(i,dic5[i])

for items in dic5.items():
    print(items)
for keys,values in dic5.items():
    print(keys,values)
```

还用我们上面的例子，存取这个班学生的信息，我们如果通过字典来完成，那：

```
dic={'zhangsan':{'age':23,'sex':'male'},
     '李四':{'age':33,'sex':'male'},
     'wangwu':{'age':27,'sex':'women'}
     }
```

八 集合(set)

集合是一个无序的，不重复的数据组合，它的主要作用如下：

- 去重，把一个列表变成集合，就自动去重了
- 关系测试，测试两组数据之前的交集、差集、并集等关系

集合(set)：把不同的元素组成一起形成集合，是python基本的数据类型。

集合元素(set elements):组成集合的成员(不可重复)

```
li=[1,2,'a','b']
s =set(li)
print(s)    # {1, 2, 'a', 'b'}

li2=[1,2,1,'a','a']
s=set(li2)
print(s)  #{1, 2, 'a'}
```

集合对象是一组无序排列的可哈希的值：集合成员可以做字典的键

```
li=[[1,2],'a','b']
s =set(li) #TypeError: unhashable type: 'list'
print(s)
```

集合分类：可变集合、不可变集合

可变集合(set)：可添加和删除元素，非可哈希的，不能用作字典的键，也不能做其他集合的元素

不可变集合(frozenset)：与上面恰恰相反

```
li=[1,'a','b']
s =set(li)
dic={s:'123'} #TypeError: unhashable type: 'set'
```

集合的相关操作

1、创建集合

由于集合没有自己的语法格式，只能通过集合的工厂方法set()和frozenset()创建

```
s1 = set('alvin')

s2= frozenset('yuan')

print(s1,type(s1))  #{'l', 'v', 'i', 'a', 'n'} <class 'set'>
print(s2,type(s2))  #frozenset({'n', 'y', 'a', 'u'}) <class 'frozenset'>
```

2、访问集合

由于集合本身是无序的，所以不能为集合创建索引或切片操作，只能循环遍历或使用in、not in来访问或判断集合元素。

```
s1 = set('alvin')
print('a' in s1)
print('b' in s1)
#s1[1]  #TypeError: 'set' object does not support indexing

for i in s1:
    print(i)
#
# True
# False
# v
# n
# l
# i
# a
```

3、更新集合

可使用以下内建方法来更新：

s.add()
s.update()

s.remove()

注意只有可变集合才能更新：

```
# s1 = frozenset('alvin')
# s1.add(0)  #AttributeError: 'frozenset' object has no attribute 'add'

s2=set('alvin')
s2.add('mm')
print(s2)  #{'mm', 'l', 'n', 'a', 'i', 'v'}

s2.update('HO')#添加多个元素
print(s2)  #{'mm', 'l', 'n', 'a', 'i', 'H', 'O', 'v'}

s2.remove('l')
print(s2)  #{'mm', 'n', 'a', 'i', 'H', 'O', 'v'}
```

del：删除集合本身

四、集合类型操作符

1 in ,not in

2 集合等价与不等价(==, !=)

3 子集、超集

```
s=set('alvinyuan')
s1=set('alvin')
print('v' in s)
print(s1<s)
```

4 联合(|)

联合(union)操作与集合的or操作其实等价的，联合符号有个等价的方法，union()。

```
s1=set('alvin')
s2=set('yuan')
s3=s1|s2
print(s3)  #{'a', 'l', 'i', 'n', 'y', 'v', 'u'}
print(s1.union(s2)) #{'a', 'l', 'i', 'n', 'y', 'v', 'u'}
```

5、交集(&)

与集合and等价，交集符号的等价方法是intersection()

```
s1=set('alvin')
s2=set('yuan')
s3=s1&s2
print(s3)  #{'n', 'a'}

print(s1.intersection(s2)) #{'n', 'a'}
```

6、查集(-)

等价方法是difference()

```
s1=set('alvin')
s2=set('yuan')
s3=s1-s2
print(s3)  #{'v', 'i', 'l'}

print(s1.difference(s2)) #{'v', 'i', 'l'}
```

7、对称差集(^)

对称差分是集合的XOR(‘异或’)，取得的元素属于s1,s2但不同时属于s1和s2.其等价方法symmetric_difference()

```
s1=set('alvin')
s2=set('yuan')
s3=s1^s2
print(s3)  #{'l', 'v', 'y', 'u', 'i'}

print(s1.symmetric_difference(s2)) #{'l', 'v', 'y', 'u', 'i'}
```

应用

```
'''最简单的去重方式'''
lis = [1,2,3,4,1,2,3,4]
print list(set(lis))    #[1, 2, 3, 4]
```

九 文件操作

9.1 对文件操作流程

1. 打开文件，得到文件句柄并赋值给一个变量
2. 通过句柄对文件进行操作
3. 关闭文件

现有文件如下：

```
昨夜寒蛩不住鸣。
惊回千里梦，已三更。
起来独自绕阶行。
人悄悄，帘外月胧明。
白首为功名，旧山松竹老，阻归程。
欲将心事付瑶琴。
知音少，弦断有谁听。
```

```
f = open('小重山') #打开文件
data=f.read()#获取文件内容
f.close() #关闭文件
```

注意 if in the win，hello文件是utf8保存的，打开文件时open函数是通过操作系统打开的文件，而win操作系统默认的是gbk编码，所以直接打开会乱码，需要f=open('**hello**',encoding='**utf8**')，hello文件如果是gbk保存的，则直接打开即可。

9.2 文件打开模式

```
========= ===============================================================
    Character Meaning
    --------- ---------------------------------------------------------------
    'r'       open for reading (default)
    'w'       open for writing, truncating the file first
    'x'       create a new file and open it for writing
    'a'       open for writing, appending to the end of the file if it exists
    'b'       binary mode
    't'       text mode (default)
    '+'       open a disk file for updating (reading and writing)
    'U'       universal newline mode (deprecated)
    ========= ===============================================================
```

先介绍三种最基本的模式：

```
# f = open('小重山2','w') #打开文件
# f = open('小重山2','a') #打开文件
# f.write('莫等闲1\n')
# f.write('白了少年头2\n')
# f.write('空悲切!3')
```

9.3 文件具体操作

操作方法介绍

```
f = open('小重山') #打开文件
# data1=f.read()#获取文件内容
# data2=f.read()#获取文件内容
#
# print(data1)
# print('...',data2)
# data=f.read(5)#获取文件内容

# data=f.readline()
```

```
# data=f.readline()
# print(f.__iter__().__next__())
# for i in range(5):
#     print(f.readline())

# data=f.readlines()

# for line in f.readlines():
#     print(line)


# 问题来了:打印所有行,另外第3行后面加上:'end 3'
# for index,line in enumerate(f.readlines()):
#     if index==2:
#         line=''.join([line.strip(),'end 3'])
#     print(line.strip())

#切记:以后我们一定都用下面这种
# count=0
# for line in f:
#     if count==3:
#         line=''.join([line.strip(),'end 3'])
#     print(line.strip())
#     count+=1

# print(f.tell())
# print(f.readline())
# print(f.tell())#tell对于英文字符就是占一个,中文字符占三个,区分与read()的不同.
# print(f.read(5))#一个中文占三个字符
# print(f.tell())
# f.seek(0)
# print(f.read(6))#read后不管是中文字符还是英文字符,都统一算一个单位,read(6),此刻就读了6个中文字符

#terminal上操作:
f = open('小重山2','w')
# f.write('hello \n')
# f.flush()
# f.write('world')

# 应用:进度条
# import time,sys
# for i in range(30):
#     sys.stdout.write("*")
#     # sys.stdout.flush()
#     time.sleep(0.1)


# f = open('小重山2','w')
# f.truncate()#全部截断
# f.truncate(5)#全部截断


# print(f.isatty())
# print(f.seekable())
# print(f.readable())

f.close() #关闭文件
```

接下来我们继续扩展文件模式:

```
# f = open('小重山2','w') #打开文件
# f = open('小重山2','a') #打开文件
# f.write('莫等闲1\n')
# f.write('白了少年头2\n')
# f.write('空悲切!3')

```

```
7
8  # f.close()
9
10 #r+,w+模式
11 # f = open('小重山2','r+') #以读写模式打开文件
12 # print(f.read(5))#可读
13 # f.write('hello')
14 # print('------')
15 # print(f.read())
16
17
18 # f = open('小重山2','w+') #以写读模式打开文件
19 # print(f.read(5))#什么都没有,因为先格式化了文本
20 # f.write('hello alex')
21 # print(f.read())#还是read不到
22 # f.seek(0)
23 # print(f.read())
24
25 #w+与a+的区别在于是否在开始覆盖整个文件
26
27
28 # ok,重点来了,我要给文本第三行后面加一行内容:'hello 岳飞!'
29 # 有同学说,前面不是做过修改了吗? 大哥,刚才是修改内容后print,现在是对文件进行修改!!!
30 # f = open('小重山2','r+') #以写读模式打开文件
31 # f.readline()
32 # f.readline()
33 # f.readline()
34 # print(f.tell())
35 # f.write('hello 岳飞')
36 # f.close()
37 # 和想的不一样,不管事!那涉及到文件修改怎么办呢?
38
39 # f_read = open('小重山','r') #以写读模式打开文件
40 # f_write = open('小重山_back','w') #以写读模式打开文件
41
42 # count=0
43 # for line in f_read:
44     # if count==3:
45     #     f_write.write('hello,岳飞\n')
46     #
47     # else:
48     #     f_write.write(line)
49
50
51     # another way:
52     # if count==3:
53     #
54     #     line='hello,岳飞2\n'
55     # f_write.write(line)
56     # count+=1
57
58
59 # #二进制模式
60 # f = open('小重山2','wb') #以二进制的形式读文件
61 # # f = open('小重山2','wb') #以二进制的形式写文件
62 # f.write('hello alvin!'.encode())#b'hello alvin!'就是一个二进制格式的数据,只是为了观看,没有显示成0101
```

注意1：无论是py2还是py3，在r+模式下都可以等量字节替换，但没有任何意义的！

注意2：有同学在这里会用readlines得到内容列表，再通过索引对相应内容进行修改，最后将列表重新写会该文件。

这种思路有一个很大的问题，数据若很大，你的内存会受不了的，而我们的方式则可以通过迭代器来优化这个过程。

补充：rb模式以及seek

在py2中：

```
#昨夜寒蛩不住鸣.

f = open('test','r',) #以写读模式打开文件

f.read(3)

# f.seek(3)
# print f.read(3) # 夜

# f.seek(3,1)
# print f.read(3) # 寒

# f.seek(-4,2)
# print f.read(3) # 鸣
```

在py3中：

```
# test:
昨夜寒蛩不住鸣.

f = open('test','rb',) #以写读模式打开文件

f.read(3)

# f.seek(3)
# print(f.read(3)) # b'\xe5\xa4\x9c'

# f.seek(3,1)
# print(f.read(3)) # b'\xe5\xaf\x92'

# f.seek(-4,2)
# print(f.read(3))    # b'\xe9\xb8\xa3'

#总结：在py3中,如果你想要字符数据,即用于观看的,则用r模式,这样我f.read到的数据是一个经过decode的
#      unicode数据； 但是如果这个数据我并不需要看,而只是用于传输,比如文件上传,那么我并不需要decode
#      直接传送bytes就好了,所以这个时候用rb模式.

#      在py3中,有一条严格的线区分着bytes和unicode,比如seek的用法,在py2和py3里都是一个个字节的seek,
#      但在py3里你就必须声明好了f的类型是rb,不允许再模糊.

#建议：以后再读写文件的时候直接用rb模式,需要decode的时候仔显示地去解码.
```

9.4 with语句

为了避免打开文件后忘记关闭，可以通过管理上下文，即：

```
with open('log','r') as f:
        pass
```

如此方式，当with代码块执行完毕时，内部会自动关闭并释放文件资源。

在Python 2.7 后，with又支持同时对多个文件的上下文进行管理，即：

```
with open('log1') as obj1, open('log2') as obj2:
    pass
```

Yuan先生
关注 - 1
粉丝 - 3924
+加关注

43 0

posted @ 2016-08-18 09:29 Yuan先生 阅读(17863) 评论(7) 编辑 收藏

Post Comment

#1楼 2016-08-26 23:59 | 龙卷风摧毁停车场 回复 引用

very good

支持(0) 反对(0)

#2楼 2017-09-06 20:59 | 翻滚吧~小强 回复 引用

非常滴不错，前来学习学习；

支持(0) 反对(0)

#3楼 2017-10-16 11:35 | xiaoyuge 回复 引用

@ vastlee
请问您的python学习目录在什么地方，根本找不到。

支持(7) 反对(0)

#4楼 2017-10-16 12:45 | 0bug 回复 引用

@ xiaoyuge
老铁 你@我干嘛呀 😅😅😅

支持(0) 反对(0)

#5楼 2017-11-15 21:29 | 成林_leon 回复 引用

细致,认真的笔记,收藏打印出来

支持(0) 反对(0)

#6楼 2018-02-06 13:28 | 向前迈一步 回复 引用

不错 学习了

支持(0) 反对(0)

#7楼 2018-02-23 20:45 | 986244073 回复 引用

细致,认真的笔记,收藏打印出来

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部