一维数据

CSV格式：

Comma-Separated Values 逗号分隔值文件格式

写个试试

```
c = ['北京', '上海', '广州', '深圳']
f = open('city.csv', 'w')
f.write(','.join(c)+'\n')
f.close()


f = open('city.csv', 'r')
c = f.read().strip('\n').split(',')
f.close()
print(c)
```

open() 函数

https://www.runoob.com/python3/python3-func-open.html

Python open() 函数用于打开一个文件，并返回文件对象，在对文件进行处理过程都需要使用到这个函数，如果该文件无法被打开，会抛出 OSError。

注意：使用 open() 函数一定要保证关闭文件对象，即调用 close() 函数。

open() 函数常用形式是接收两个参数：文件名(file)和模式(mode)。

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,
newline=None, closefd=True, opener=None)
```

常用Mode:

r 只读，文件指针指向开头

r+ 打开一个文件用于读写。文件指针将会放在文件的开头。

w 打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。

w+ 打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。

a 打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。

总结:

r 读

w 写

+ 增加同时读写的功能

b 二进制(一般用于非文本)

a 追加

mode 以 w 打开文件时，文件不存在则创建，以 r 时， 不存在则报错

```
test1:
f = open('test.txt', mode='w')
print('文件的名字：', f.name)
f.write("小博士")
f.close()
print('################################')
# f = open('test.txt', mode='w')
f = open('test.txt', mode='a')
print('文件的名字：', f.name)
f.write("小博士python")
f.close()
print('################################')
f = open('test.txt', mode='r')
str = f.read()
print(str)
f.close()

test2:  这样是不行的，不会打印文档的内容
f = open('test.txt', mode='w+')
print('文件的名字：', f.name)
f.write("小博士")
str = f.read()
print(str)
f.close()

为什么呢？指针
```

write & read 方法

write() 方法用于向文件中写入指定字符串。

在文件关闭前或缓冲区刷新前，字符串内容存储在缓冲区中，这时你在文件中是看不到写入的内容的。

```
fileObject.write(str)
str -- 要写入文件的字符串。
返回的是写入的字符长度。
```

read() 方法用于从文件读取指定的字节数，如果未给定或为负则读取所有。

```
fileObject.read([size]);
size -- 从文件中读取的字节数，默认为 -1，表示读取整个文件。
返回从字符串中读取的字节。
```

拓展：

<file>.readline(size = -1) 从文件读入一行内容，如果给出参数，读入该行前size长度的字符串或者字节流

<file>.readlines(hint = -1) 从文件中读入所有行，以每行为元素形成一个列表，如果给出参数，读入hint行

```
f = open('test.txt', mode='r')
print('文件的名字：', f.name)
a = f.readline()
print(a)
f.close()

f = open('test.txt', mode='r')
print('文件的名字：', f.name)
a = f.readlines()
print(a)
f.close()
```

<file>.writeline(lines) 将一个元素全为字符串的列表写入文件

<file>.seek(offset[, whence])

- **offset** -- 开始的偏移量，也就是代表需要移动偏移的字节数，如果是负数表示从倒数第几位开始。
- **whence**：可选，默认值为 0。给 offset 定义一个参数，表示要从哪个位置开始偏移；0 代表从文件开头开始算起，1 代表从当前位置开始算起，2 代表从文件末尾算起。

```
f = open('test.txt', mode='w+')
```

```
print('文件的名字：', f.name)
l = ['芳宁', '佳宸']
f.write(l)
f.close()

f = open('test.txt', mode='w+')
print('文件的名字：', f.name)
l = ['芳宁', '佳宸']
f.writelines(l)
f.seek(0)
a = f.read()
print(a)
f.close()
```

f.write(','.join(c)+'\n')

Python join() 方法用于将序列中的元素以指定的字符连接生成一个新的字符串。

```
str.join(sequence)
sequence -- 要连接的元素序列。
返回通过指定字符连接序列中元素后生成的新字符串。
```

+'\n' 作用是在字符串的结尾换行。

c = f.read().strip('\n').split(',')

Python strip() 方法用于移除字符串头尾指定的字符（默认为空格）或字符序列。

注意：该方法只能删除开头或是结尾的字符，不能删除中间部分的字符。

```
str.strip([chars]);
chars -- 移除字符串头尾指定的字符序列。
返回移除字符串头尾指定的字符序列生成的新字符串。
```

split() 通过指定分隔符对字符串进行切片，如果第二个参数 num 有指定值，则分割为 num+1 个子字符串。

```
str.split(str="", num=string.count(str))
str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。
num -- 分割次数。默认为 -1, 即分隔所有。
返回分割后的字符串列表。
```

通过文件读写命令复制一张图片

```
with open('photo2.png','rb') as file:  # 以“rb”模式打开图片
    data = file.read()
    with open('photo3.png','wb') as newfile:  # 以“wb”模式写入
        newfile.write(data)
```

with open() as f 的用法：

由于文件读写时有可能产生`IOError`，一旦出错，后面的`f.close()`就不会调用。所以，为了保证无论是否出错都能正确地关闭文件，我们可以使用`try ... finally`通过捕捉异常、处理异常来实现。

如果能保证文件打开没有异常的情况下，我们每次都这么写，实在太繁琐。所以，Python引入了`with`语句来自动帮我们调用`close()`方法。也就是说：文件读取的这个操作，只有在`with`语句内部才会生效，不会离开。

```
with open(文件名, 模式) as 文件对象:
    文件对象.方法()

with open('test.txt', 'r') as f:
    print(f.read())
```

作业：

帮助语文老师做一个古诗自动挖空程序，将课本中的古诗写入txt中， 程序读取txt并随机挖空，让使用者填写。

拓展：输入姓名，判断正确错误，记录成绩。

参考程序：

```
list_test = ['一弦一柱思华年。\n','只是当时已惘然。\n']  # 将要默写的诗句放在列表里。
with open ('test.txt','r') as f:
    lines = f.readlines()
print(lines)
with open('test.txt','w') as new:
    for line in lines:
        if line in list_test:  # 属于默写列表中的句子，将其替换成横线。
            new.write('_____________。\n')
        else:
            new.write(line)
```

另外问问家长，看看工作中有无程序可以替代的部分，我们试着编程解决。

未完。。。。

(15条消息) python读写文件练习_Crystal_LYP的博客-CSDN博客

回顾

10/15

回顾上节课讲了什么知识

打开一个文件用什么函数?

通常打开文件和什么是成对出现的?

打开文件的函数的参数是什么?

打开文件的"模式"可以有那些，有什么区别?

读写文件时光标默认的位置在哪里?

如何实现读写文件时光标在文件的末尾?

csv格式是怎样的一种格式?

下面继续课程。。。。

seek

```
f = open("./workfile.txt","w+")
f.write("12345ssdlg")
f.seek(3, 0)  #f.seek(-3, 2)
a = f.read(1)
print(a)
f.close()
```

`io.UnsupportedOperation: can't do nonzero end-relative seeks` 这个错误，主要是因为在python3和python2的问题，如果该程序在Python2中是不会报错的，Python3则会报错。因为Pyhon3在文本文件中，没有使用b模式选项打开的文件，只允许从文件头开始计算相对位置，从文件尾计算时就会引发异常

```
f = open("./workfile.txt","rb")
f.seek(-3, 2)
a = f.read(1)
```

```
print(a)
f.close()
不是太重要
我们知道seek可以改变光标位置就好
```

readline

readlines

writeline

strip

join

再看

```
c = ['北京', '上海', '广州', '深圳']
f = open('city.csv', 'w')
f.write(','.join(c)+'\n')
f.close()


f = open('city.csv', 'r')
c = f.read().strip('\n').split(',')
f.close()
print(c)
```

with open