1. **Python的文件处理**

**第一部分**

**一、读取一个文件**

**1.1、read()方法**

创建一个名叫file.txt,里面内容是“this is a text file,content is Hello python!我喜欢Python!”

#打开一个文件,文件名是file.txt,模式是"只读"

f=open("file.txt",mode="rb")

#读取文件，decode("gb2312")对内容进行编码，解决中文乱码问题

text=f.read().decode("gb2312")

print(text)

f.close()

输出结果：

this is a text file,content is Hello python!我喜欢Python!

read()方法中参数用于设置读取文件的大小，需要注意的是，一个英文占一个字节，一个汉字占两个字节。比如下列：

#打开一个文件,文件名是file.txt,模式是"只读",rb读取一个二进制文件。

f=open("file.txt",mode="r")

#读取文件，decode("gb2312")对内容进行编码，解决中文乱码问题

text=f.read(20).decode("gb2312")

print(text)

f.close()

输出结果：

this is a text file,

**1.2、如何解决中文乱码问题：**

1、我们在处理文本文件的时候，如果该文件是unicode编码，则不需要做任何的处理操作，直接用'r'参数读取直接可用。

2、如果不是unicode编码，就不能直接用了,直接读会出现乱码。只能先以'rb'参数读取二进制文件的方式读取进来，read之后再解码。

f=open('文件路径','rb')

f\_read=f.read()

f\_read\_decode=f\_read.decode('该文件的编码方式')

print(f\_read\_decode)

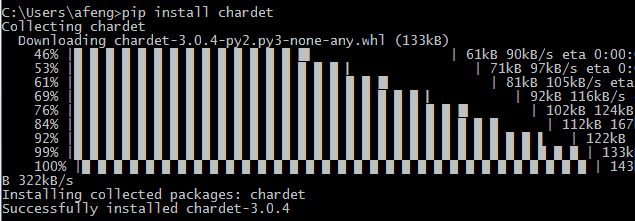
f.close()

　　这样才能完整显示，不然会有乱码出现。

　　3、问题又来了，往往我们并不知道该文件的编码方式，这该怎么办？

解决方案如下：

3.1、先使用pip命名安装chardet



3.2、使用如下代码获取文件的编码方式

import chardet

f=open("file.txt",mode="rb")

f\_read=f.read()

f\_charInfo=chardet.detect(f\_read)

print(f\_charInfo)

执行结果：

{'confidence': 0.99, 'language': 'Chinese', 'encoding': 'GB2312'}

上面文件的编码方式是GB2312

3.3、指定解码是文件的编码方式即可

import chardet

f=open("file.txt",mode="rb")

f\_read=f.read().decode("gb2312")

print(f\_read)

f.close()

**1.3 readline()**

f.readline() 会从文件中读取单独的一行，f.readline() 如果返回一个空字符串, 说明已经已经读取到最后一行。如果设置参数为n,则每行读取n个字符。

例子1：

f=open("file.txt",mode="rb")

#f\_read=f.readline().decode("gb2312")

f\_read=f.readline()

print(f\_read)

f.close()

执行结果：

I LOVE python1!

例子2：

使用readline()方法读取整个文件

f=open("file.txt",mode="r")

f\_read=f.readline()

while(f\_read!=""):

print(f\_read)

f\_read=f.readline()

f.close()

**第二部分**

**1.4 readlines()**

f.readlines() 将返回该文件中包含的所有行，返回值为一个列表，如果设置可选参数 sizehint, 则读取指定长度的字节, 并且将这些字节按行分割，但是sizehint只是在Python3中有效。

f=open("file.txt",mode="r")

f\_read=f.readlines()

print(type(f\_read))

print(f\_read)

f.close()

<type 'list'>

['I LOVE python1!\n', 'I LOVE python2!\n', 'I LOVE python3!\n', 'I LOVE python4!\n', 'I LOVE python5!\n', 'I LOVE python6!']

f=open("file.txt",mode="r")

f\_read=f.readlines(15)

print(type(f\_read))

print(f\_read)

f.close()

执行结果：



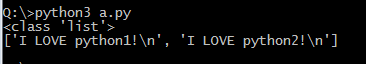
f=open("file.txt",mode="r")

f\_read=f.readlines(16)

print(type(f\_read))

print(f\_read)

f.close()

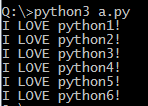


**1.5 使用for语句读取文件**

f=open("file.txt",mode="r")

for line in f.read():

print(line,end="")



**二、写内容到一个文件**

设置模式方式为w,方法为write()

1、原来文本上的内容都会被删除

2、如果原文件不存在，则会自动创建一个文件

3、如果要写入一些不是字符串的东西, 那么将需要先进行转换

例子:

#指定要写的文件名和模式

f=open("file2.txt",mode="w")

#写入文件的内容

str="you love python too!"

f.write(str)

#关闭文件

f.close()

**第三部分**

**三、在文件中追加内容**

设置模式方式为a,方法为write()

1、在原来文本上的内容追加内容

2、如果原文件不存在，则会自动创建一个文件

3、如果要写入一些不是字符串的东西, 那么将需要先进行转换

#指定要写的文件名和模式

f=open("file3.txt",mode="a")

#写入文件的内容

str="我喜欢Python2!\n"

f.write(str)

#关闭文件

f.close()

**四、其他常见文件操作方法**

1、file.tell()

f.tell() 返回文件对象当前所处的位置, 它是从文件开头开始算起的字节数。

2、f.seek()

如果要改变文件当前的位置, 可以使用 f.seek(offset, from\_what) 函数。

from\_what 的值, 如果是 0 表示开头, 如果是 1 表示当前位置, 2 表示文件的结尾，例如：

 seek(x,0) ： 从起始位置即文件首行首字符开始移动 x 个字符

 seek(x,1) ： 表示从当前位置往后移动x个字符

 seek(-x,2)：表示从文件的结尾往前移动x个字符

3、f.close()在文本文件中 (那些打开文件的模式下没有 b 的), 只会相对于文件起始位置进行定位。

当你处理完一个文件后, 调用 f.close() 来关闭文件并释放系统的资源，如果尝试再调用该文件，则会抛出异常。

4、[file.flush()](http://www.runoob.com/python3/python3-file-flush.html)

刷新文件内部缓冲，直接把内部缓冲区的数据立刻写入文件, 而不是被动的等待输出缓冲区写入。

5、[file.isatty()](http://www.runoob.com/python3/python3-file-isatty.html)

如果文件连接到一个终端设备返回 True，否则返回 False。

6、[file.next()](http://www.runoob.com/python3/python3-file-next.html)

返回文件下一行。

7、[file.truncate([size])](http://www.runoob.com/python3/python3-file-truncate.html)

从文件的首行首字符开始截断，截断文件为 size 个字符，无 size 表示从当前位置截断；截断之后 V 后面的所有字符被删除，其中 Widnows 系统下的换行代表2个字符大小。

**五、不同模式打开文件的完全列表**

常用打开模式：

r 只能读   
r+ 可读可写，不会创建不存在的文件，从顶部开始写，会覆盖之前此位置的内容  
w+ 可读可写，如果文件存在，则覆盖整个文件，不存在则创建  
w 只能写，覆盖整个文件，不存在则创建   
a 只能写，从文件底部添加内容 不存在则创建   
a+ 可读可写 从文件顶部读取内容 从文件底部添加内容 不存在则创建

|  |  |
| --- | --- |
| 模式 | 描述 |
| r | 以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。 |
| rb | 以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。 |
| r+ | 打开一个文件用于读写。文件指针将会放在文件的开头。 |
| rb+ | 以二进制格式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将其覆盖。如果该文件不存在，创建新文件。 |
| wb | 以二进制格式打开一个文件只用于写入。如果该文件已存在则将其覆盖。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将其覆盖。如果该文件不存在，创建新文件。 |
| wb+ | 以二进制格式打开一个文件用于读写。如果该文件已存在则将其覆盖。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。 |
| ab | 以二进制格式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。 |
| a+ | 打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该  文件不存在，创建新文件用于读写。 |
| ab+ | 以二进制格式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。如果该文件不  存在，创建新文件用于读写。 |

开发实例：对同一个文件，先读后写

#指定要写的文件名和模式

f=open("file5.txt",mode="a+")

#写入文件的内容

str="I love Python2!\n"

f.write(str)

f.flush()

f.seek(0)

str2=f.read()

print(str2)

#关闭文件

f.close()

**第四部分**

1. **序列化和反序列化**

6.1序列化 (serialization)  
序列化是将对象状态转换为可保持或传输的格式的过程。与序列化相对的是反序列化，它将流转换为对象。这两个过程结合起来，可以轻松地存储和传输数据。

6.2序列化和反序列化的目的　　  
1、以某种存储形式使自定义对象持久化；　　  
2、将对象从一个地方传递到另一个地方。　　  
3、使程序更具维护性

6.3 序列化

  由于存在于内存中的对象都是暂时的，无法长期驻存，为了把对象的状态保持下来，这时需要把对象写入到磁盘或者其他介质中，这个过程就叫做序列化。

将变量或对象通过序列化，转化成字节流（bytes）存储到文件中（磁盘）或者通过网络传输到需要的机器上。

因此，我们把变量或对象从内存中变成可存储或传输的过程称之为序列化

6.4 反序列化

  反序列化恰恰是序列化的反向操作，也就是说，把已存在在磁盘或者其他介质中的对象，反序列化（读取）到内存中，以便后续操作，而这个过程就叫做反序列化。

  概括性来说序列化是指将对象实例的状态存储到存储媒体（磁盘或者其他介质）的过程。在此过程中，先将对象的公共字段和私有字段以及类的名称（包括类所在的程序集）转换为字节流，然后再把字节流写入数据流。在随后对对象进行反序列化时，将创建出与原对象完全相同的副本。

**七、pickle 模块**

python的pickle模块实现了基本的数据序列和反序列化。

通过pickle模块的序列化操作我们能够将程序中运行的对象信息保存到文件中去，永久存储。

通过pickle模块的反序列化操作，我们能够从文件中创建上一次程序保存的对象。

7**.1 基本接口：**

1、pickle.dump(obj, file, [,protocol])  
　　注解：将对象obj保存到文件file中去。  
　　protocol为序列化使用的协议版本，0：ASCII协议，序列化的对象使用可打印的ASCII码表示；1：老式的二进制协议；2：2.3版本引入的新二进制协议，较以前的更高效。其中协议0和1兼容老版本的python。protocol默认值为0。  
　　file：对象保存到的类文件对象。

2、pickle.load(file)  
　　注解：从file中读取一个字符串，并将它重构为原来的python对象。  
　　file:类文件对象

**7.2 开发实例1**

#导入Python中用于序列和反序列的pickle模块

import pickle

#pprint模块 提供了打印出任何python数据结构类和方法

import pprint

#实现数据对象的序列操作

#定义一个Python对象的数据

data1 = {'a': [1, 2.0, 3, 4+6j],

'b': ('string', u'Unicode string'),

'c': None}

#打开一个文件，模式是写二进制的写模式

output = open('data.pkl', 'wb')

#将对象obj保存到文件file中去。

pickle.dump(data1, output)

#关闭文件

output.close()

#实现数据对象的反序列操作

#打开一个文件，模式是写二进制的读模式

pkl\_file = open('data.pkl', 'rb')

#从file中读取一个字符串，并将它重构为原来的python对象

data1 = pickle.load(pkl\_file)

#打印出这个对象

pprint.pprint(data1)

#关闭文件

pkl\_file.close()

**八、with语句**

with 语句适用于对资源进行访问的场合，确保不管使用过程中是否发生异常都会执行必要的“清理”操作，释放资源，比如文件使用后自动关闭、线程中锁的自动获取和释放等。

with open("myfile.txt") as f:

for line in f:

print(line, end="")

**九、开发实例**

1、获取文件的后缀名

def getFilename\_extension(filename):

return filename[filename.rfind('.')+1:]

print(getFilename\_extension("awe.23.txt"))

#coding=utf-8

2、爬取页面内容，并写入指定文件。

**import** urllib.request  
fi = open(**"rupeng.txt"**, **'wb'**)  
fi.write(urllib.request.urlopen(**'http://www.rupeng.com/'**).read())  
fi.close()

3、读取文件的内容，把出现a的内容全部变为A

f=open("file5.txt",mode="r+")

str2=f.read()

print(str2)

f.flush()

f.seek(0)

#写入文件的内容

str2=str2.replace("a", "A")

f.write(str2)

#关闭文件

f.close()

方法2：

#coding=utf-8

f=open("file5.txt",mode="r+")

lines=f.readlines()

i=1;

for line in lines:

line=line.replace("a","A")

f.seek(len(line\*(i-1)))

f.write(line)

i=i+1

#关闭文件

f.close()