**python3中StringIO和BytesIO使用方法和使用场景详解**

说起IO，很多人首先想到的是磁盘中的文件，将磁盘中的文件读到[内存](https://so.csdn.net/so/search?q=%E5%86%85%E5%AD%98&spm=1001.2101.3001.7020)以及内存内容写入文件。但是还有一种内存和内存之间的IO，叫类文件对象，这一篇我们就一起来学习下python中的两个类文件对象：StringIO和BytesIO。

我是T型人小付，一位坚持终身学习的互联网从业者。喜欢我的博客欢迎在csdn上关注我，如果有问题欢迎在底下的评论区交流，谢谢。

**内存中的IO**

首先必须要搞清一个问题，就是为什么要有内存级别的IO？

之前说的磁盘上的文件，就是将数据持久化到磁盘的一块区域，供后面重复使用。其优点就是持久化稳定不丢失，但是缺点也很明显，就是每次要使用都要从磁盘读入，相对内存而言很缓慢。

如果只是短时间的重复利用，并不希望长期持久化，而且对速度的要求比较高，这时候就可以考虑缓存。说到缓存，很多朋友就想到redis，熟悉python的朋友还会想到装饰器和闭包函数。

不过python已经原生为我们准备好了类文件对象（file-like object），这种对象在内存中创建，可以像文件一样被操作。

下面我们就来学习下两种类文件对象，StringIO和BytesIO。

**操作环境**

因为python3中将StringIO和BytesIO都集成到了io模块中，导入方式和python2不同，这个要注意。

**标志位**

内存中的对象有一个标志位的概念，往里面写入，标志位后移到下一个空白处。而读数据的时候是从标志位开始读，所以想要读取前面的数据需要手动将标志位进行移动。在下面的操作中我们会看到。

**StringIO**

看名字就大概能猜到，这种类文件对象是用来存储字符串的。

新建一个StringIO对象

In [1]: from io import StringIO

In [2]: s=StringIO()

In [3]: type(s)

Out[3]: \_io.StringIO

写入一些字符串

In [6]: s.write('this\nis\na\ngreat\nworld!')

Out[6]: 22

In [7]: s.read()

Out[7]: ''

可以看到尝试读取的时候返回为空，这就是上面提到的标志位的原因。因为此时write以后标志位跑到了第23字节，所以再往后读取就是空值。

将标志位移动到最前面再尝试读取，就能成功了

In [11]: s.seek(0)

Out[11]: 0

In [12]: s.read()

Out[12]: 'this\nis\na\ngreat\nworld!'

但是获取全部值方法不受这个标志位影响

In [13]: s.read()

Out[13]: ''

In [14]: s.getvalue()

Out[14]: 'this\nis\na\ngreat\nworld!'

除了一次读取全部内容，还可以\n为分界读取单行

In [15]: s.seek(0)

Out[15]: 0

In [16]: s.readline()

Out[16]: 'this\n'

In [17]: s.readline().strip()

Out[17]: 'is'

In [18]: s.seek(0)

Out[18]: 0

In [19]: s.readlines()

Out[19]: ['this\n', 'is\n', 'a\n', 'great\n', 'world!']

该对象使用完毕直接关掉，该内存里的内容被清空

In [20]: s.close()

**BytesIO**

StringIO只能存储字符串，遇到从网络下载的图片视频等Bytes类型的内容就不行了，需要用到专门存储Bytes类型的BytesIO对象。

其用法和StringIO大同小异。

新建一个对象，写入一些Bytes数据

In [1]: from io import BytesIO

In [2]: b=BytesIO()

In [3]: b.write('小付'.encode('utf-8'))

Out[3]: 6

In [4]: b.getvalue()

Out[4]: b'\xe5\xb0\x8f\xe4\xbb\x98'

BytesIO真正实用的地方还是在于存储图片视频等数据，不管是本地生成的还是网络下载的。

利用requests库从网络下载一张图片

In [7]: response=requests.get('https://img.zcool.cn/community/0170cb554b9200000001bf723782e6.jpg@1280w\_1l\_2o\_100sh.jpg')

In [8]: type(response.content)

Out[8]: bytes

因为是Bytes类型的，就可以直接保存到BytesIO中

In [12]: img=BytesIO(response.content)

想要对图片进一步处理这里使用Python3的Pillow库，需要注意的是这里下载的是pillow，但是导入的时候还是从PIL导入，这是一个历史遗留问题。

In [13]: from PIL import Image

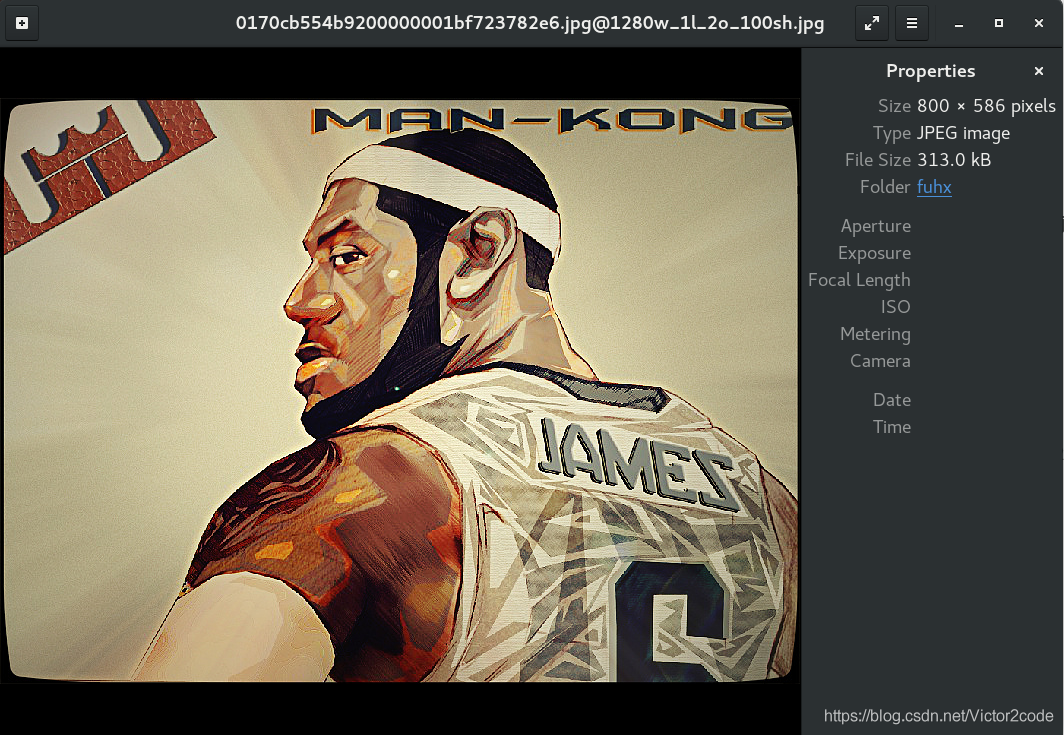
In [14]: pic=Image.open(img)

In [15]: pic.format

Out[15]: 'JPEG'

In [16]: pic.size

Out[16]: (800, 586)

这些信息和图片本身是一样的  
  
可能有朋友会问，我直接把下载的Bytes类型用Image打开不行吗？

答案是不可以，Image接受的参数是一个文件对象，或者类文件对象。所以要么是磁盘上的文件，要么是内存中的BytesIO。

**总结**

了解了StringIO和BytesIO这两个内存中的类文件对象，一些不必永久存储在磁盘上的临时文件就可以直接放内存中使用了，不过和文件一样，使用完记得及时关闭回收内存空间。