open函数

open函数

open 函数可以打开一个文件。超级简单吧？大多数时候，我们看到它这样被使用：

```
f = open('photo.jpg', 'r+')
jpgdata = f.read()
f.close()
```

我现在写这篇文章的原因，是大部分时间我看到open被这样使用。有三个错误存在于上面的代码中。你能把它们全指出来吗？如不能，请读下去。在这篇文章的结尾，你会知道上面的代码错在哪里，而且，更重要的是，你能在自己的代码里避免这些错误。现在我们从基础开始：

open的返回值是一个文件句柄，从操作系统托付给你的Python程序。一旦你处理完文件，你会想要归还这个文件句柄，只有这样你的程序不会超出一次能打开的文件句柄的数量上限。

显式地调用close关闭了这个文件句柄，但前提是只有在read成功的情况下。如果有任意异常正好在`f = open(...)`之后产生，`f.close()`将不会被调用（取决于Python解释器的做法，文件句柄可能还是会被归还，但那是另外的话题了）。为了确保不管异常是否触发，文件都能关闭，我们将其包裹成一个with语句:

```
with open('photo.jpg', 'r+') as f:
    jpgdata = f.read()
```

open的第一个参数是文件名。第二个(mode 打开模式)决定了这个文件如何被打开。

- 如果你想读取文件，传入r
- 如果你想读取并写入文件，传入r+
- 如果你想覆盖写入文件，传入w
- 如果你想在文件末尾附加内容，传入a

虽然有若干个其他的有效的mode字符串，但有可能你将永远不会使用它们。mode很重要，不仅因为它改变了行为，而且它可能导致权限错误。举个例子，我们要是在一个写保护的目录里打开一个jpg文件，`open(.., 'r+')`就失败了。mode可能包含一个扩展字符；让我们还可以以二进制方式打开文件(你将得到字节串)或者文本模式(字符串)

一般来说，如果文件格式是由人写的，那么它更可能是文本模式。jpg图像文件一般不是人写的（而且其实不是人直接可读的），因此你应该以二进制模式来打开它们，方法是在mode字符串后加一个b(你可以看看开头的例子里，正确的方式应该是rb)。
如果你以文本模式打开一些东西（比如，加一个t,或者就用r/r+/w/a），你还必须知道要使用哪种编码。对于计算机来说，所有的文件都是字节，而不是字符。

可惜，在Pyhon 2.x版本里，open不支持显示地指定编码。然而，io.open函数在Python 2.x中和3.x(其中它是open的别名)中都有提供，它能做正确的事。你可以传入encoding这个关键字参数来传入编码。
如果你不传入任意编码，一个系统 - 以及Python -指定的默认选项将被选中。你也许被诱惑去依赖这个默认选项，但这个默认选项经常是错误的，或者默认编码实际上不能表达文件里的所有字符（这将经常发生在Python 2.x和/或Windows）。

所以去挑选一个编码吧。utf-8是一个非常好的编码。当你写入一个文件，你可以选一个你喜欢的编码（或者最终读你文件的程序所喜欢的编码）。

那你怎么找出正在读的文件是用哪种编码写的呢？好吧，不幸的是，并没有一个十分简单的方式来检测编码。在不同的编码中，同样的字节可以表示不同，但同样有效的字符。因此，你必须依赖一个元数据（比如，在HTTP头信息里）来找出编码。越来越多的是，文件格式将编码定义成UTF-8。

有了这些基础知识，我们来写一个程序，读取一个文件，检测它是否是JPG（提示：这些文件头部以字节FF D8开始），把对输入文件的描述写入一个文本文件。

```
import io

with open('photo.jpg', 'rb') as inf:
    jpgdata = inf.read()

if jpgdata.startswith(b'\xff\xd8'):
    text = u'This is a JPEG file (%d bytes long)\n'
else:
    text = u'This is a random file (%d bytes long)\n'

with io.open('summary.txt', 'w', encoding='utf-8') as outf:
    outf.write(text % len(jpgdata))
```

我敢肯定，现在你会正确地使用open啦！