

---

# Table of Contents

前言	1.1
第零周课程问题集	1.2
Python 基础问题集	1.2.1
Python 对象问题	1.2.1.1
代码缩进引起的错误	1.2.1.2
String 类型使用 split()分割	1.2.1.3
文件名跟模块名字冲突错误	1.2.1.4
with as 的用法	1.2.1.5
中文 url 编码问题	1.2.1.6
内置函数 zip( )的使用	1.2.1.7
Python 读写 CSV文件	1.2.1.8
写路径问题	1.2.1.9
python简单的排错技巧	1.2.1.10
PyCharm 编辑器问题集	1.2.2
安装库之后，运行提示没有库	1.2.2.1
第一周课程问题集	1.3
BeautifulSoup4问题集	1.3.1
错误提示锦集	1.3.1.1
soup.select ( )选择标签问题	1.3.1.2
编码输出问题	1.3.1.3
get_text 和 get()的使用	1.3.1.4
find_all()和 select()的区别	1.3.1.5
soup查找tag，结果为空[]	1.3.1.6
soup.select()尽量不使用完整selector	1.3.1.7
pip 问题集	1.3.2
pip 怎么更新 Python 包？	1.3.2.1
代理ip问题集	1.3.3
如何获取代理ip	1.3.3.1
代理ip不能使用，导致爬虫不能运行	1.3.3.2
js加载问题	1.3.4

---

爬虫遇js加载解决办法一	1.3.4.1
第二周课程问题集	1.4
MongoDB 问题集	1.4.1
怎么导入.csv和.json文件数据	1.4.1.1
安装MongoDB出现的错误集合	1.4.1.2
爬虫报错问题集	1.4.2
断点详解	1.4.2.1
图片下载乱码	1.4.2.2
多进程问题集	1.4.3
pool线程池的使用	1.4.3.1
第三周课程问题集	1.5
第四周课程问题集	1.6
Django 问题集	1.6.1
重设 Admin 密码	1.6.1.1
关于django数据流流程问题	1.6.1.2
Django数据库连接问题	1.6.1.3
Semantic UI 问题集	1.6.2
semantic UI 那些文件在哪里？	1.6.2.1
课程相关的实用资源	1.7

---

# 前言

## 起源

我们发现市面上没有一门课程能帮新手快速上手实战项目，于是设计了这样一门课程，循序渐进的提升你的实战能力。学员这样评价课程

「这是唯一能看懂的编程课，感觉和看美剧似的，会上瘾。」



- 实战课程地址：<http://study.163.com/course/courseMain.htm?courseId=1002794001>
- 课程源码：<https://github.com/mugglecoding>

---

## 这篇文档的作用

同学们在课程的学习过程中会遇到各种类型的问题，而这篇文档的作用就是把各种问题归类并作出解答，方便同学们阅读、学习。

---

## 文档阅读指南

爬虫实战计划课程主要针对以下几个主题：

- 第零周：Python 语言基础使用
- 第一周：基本的网页爬取
- 第二周：多线程爬虫，MongoDB 数据库使用
- 第三周：数据统计和分析，可视化输出

- 第四周：Django 框架的使用

所以，问题的归类也会涉及其中的主题。除此之外，由于很多同学是编程新手，在编程环境配置，终端命令操作，编辑器使用，插件工具安装方面也会遇到不少问题，在归类当中也会涉及。

解答问题的内容，形式如下：

#### 问题描述

这一部分内容是描述同学们会问的问题，尽量还原问题原文。

#### 问题解答

这一部分内容是我通过自己的经验，理解，翻阅文档整理出来的答案。

#### 问题衍生

这一部分内容是对问题进行比较深层次的扩展，目的在于对于问题的本质有一个更深的理解，而且还会尽量引导大家怎么独立去解决一个问题。

## 答疑团队

这份文档由答疑团队持续维护：

答疑团队	AJ Kipper	志怀
简介	耐心细致，喜欢探索新事物。兴趣广泛，爱玩摇滚	喜欢用编程解决生活中的问题
Github/blog	<a href="https://github.com/AJKipper">https://github.com/AJKipper</a>	tor1024.com
邮箱	xprobie@gmail.com	717191517@qq.com
QQ	517450974	717191517

## 第零周课程问题集

# Python基础问题集

## 内容概括

这个问题集涉及Python语言本身的问题，可能是Python2.x和Python3.x之间的差别问题，可能是模块的使用问题，可能是语法的问题，范围很广。

这方面问题的产生原因，主要是同学们对Python基础知识不够牢固，所以多阅读此类问题集有助于写出更好的Python程序。

以下是Python基础的有关资料，可以自行前往。

## 参考资料

1. [魔力手册for实战学员预习.pdf](#)
2. [Python 3 教程|菜鸟教程](#)

# Python对象问题

## 问题描述：

老师，python不是面向对象的吗，这个结果怎么为true？

同学的代码

```
a = {1,2,3}
b = {1,2,3,4,5} - {4,5}
print (a == b)
#True
```

## 问题解答：

首先这个问题跟Python是否是面向对象是没有关系的，面向对象编程是一种思想，Python是面向对象的编程语言，但跟这个问题无关。

Python的对象包含着3个要素：

- **id**：用来唯一标识对象，就像身份证号码代表唯一的你。
- **type**：标识对象的类型，比如，整数型int。
- **value**：对象的值。

而对象的名字只是对象的引用，言外之意就是说同一个对象（object）可以有不同的名字（引用），就比如你有身份证的名字，但也有小时候的乳名。

所以这位同学问题中 `a == b` 的值是True是没有问题的，但是 `a == b` 的意思是a的值(value)等于b，而不是说a对象就是b对象。在Python中，有一个 `is` 关键字来判断两个对象是否是同一个对象(也就是说id是否一样)，或者用内置函数 `id()` 来输出对象的id值判断，如下：

```
>>> a = {1,2,3}
>>> b = {1,2,3,4,5} - {4,5}
>>> print (a == b)
True
>>> print (a is b)
False
>>> print (id(a),id(b))
4338869416 4338870760
```

可见，a与b的值相等，id不相等，问题也就弄清楚了。

## 问题延伸：

先看以下代码：

```
>>> a = 1
>>> b = 5 - 4
>>> a is b
True
>>> a == b
True
>>> id(a), id(b)
(4297537792, 4297537792)
```

为什么这种情况下，a与b的value和id都相等呢？

这就要回溯到Python源代码(C语言代码)本身了。Python中整数1，2，3等都是对象，而且会被引用得特别多，为了提高Python的运行效率，引入了整数对象池的机制。你不用理解什么是整数对象池机制，你只要记住，当整数对象数值相等，而且大小在[-5, 256)范围内，那就为这个整数值只开辟一个内存地址(id)，超过这个范围，则重新开辟内存地址，如下(整数数值在[-5,256)之外)：

```
>>> a = -6
>>> b = -6
>>> a is b
False
>>> a == b
True
>>> id(a), id(b)
(4338513008, 4338513040)
```

对于string对象，源码实现有一个操作叫intern，在新建string对象时会校验其内容，如果在intern池内有一样的就会把引用指过去，所以id也是一样的，如下：

```
>>> a = '1234567'
>>> b = '1234567'
>>> a is b
True
>>> a == b
True
>>> id(a), id(b)
(4338891608, 4338891608)
```

对float对象，由于float太多了，所以每一次创建对象都开辟新内存(id)。还有，对于dict，list和tuple对象都是如此，如下：



```
>>> a = 1.1
>>> b = 1.1
>>> a is b
False
>>> a = [1]
>>> b = [1]
>>> a is b
False
>>> a = (1,)
>>> b = (1,)
>>> a is b
False
>>> a = {}
>>> b = {}
>>> a is b
False
```

## 代码缩进引起的错误

### 问题描述:

老师你好，我按照视频写的代码没有输出，请问是什么原因？

同学的代码

```
from bs4 import BeautifulSoup

info = []
with open('/home/ee/Documents/1_2code_of_video/web/new_index.html', 'r') as wb_data:
    Soup = BeautifulSoup(wb_data, "lxml")
    images = Soup.select('body > div.main-content > ul > li > img')
    titles = Soup.select('body > div.main-content > ul > li > div.article-info > h3 > a')
    desc = Soup.select('body > div.main-content > ul > li > div.article-info > p.description')
    rates = Soup.select('body > div.main-content > ul > li > div.rate > span')
    cate = Soup.select('body > div.main-content > ul > li > div.article-info > p.meta-info')

for image,title,desc,rate,cate in zip(images,titles,descs,rates,cates):
    data = {
        'image': image.get('src'),
        'title': title.get_text(),
        'desc' : desc.get_text(),
        'rate' : rate.get_text(),
        'cate' : list(cate.stripped_strings)
    }

    info.append(data)

for i in info:
    if float(i['rate'])>3:
        print(i['title'],i['cate'])
```

下面是运行结果：

```
/usr/bin/python3.5 /home/ee/Documents/1_2code_of_video/web/work2.py
Process finished with exit code 0
```

### 问题解答：

同学，你出现在缩进问题上，缩进不对，对代码逻辑就有影响了。比如我先改改你的代码，先不判断分数是否大于3，我先看看info[]里面总共都有什么，如下：

```
from bs4 import BeautifulSoup

info = []
with open('/home/ee/Documents/1_2code_of_video/web/new_index.html', 'r') as wb_data:
    Soup = BeautifulSoup(wb_data, "lxml")
    images = Soup.select('body > div.main-content > ul > li > img')
    titles = Soup.select('body > div.main-content > ul > li > div.article-info > h3 > a')
    desc = Soup.select('body > div.main-content > ul > li > div.article-info > p.description')
    rates = Soup.select('body > div.main-content > ul > li > div.rate > span')
    cates = Soup.select('body > div.main-content > ul > li > div.article-info > p.meta-info')

for image,title,desc,rate,cate in zip(images,titles,descs,rates,cates):
    data = {
        'image': image.get('src'),
        'title': title.get_text(),
        'desc' : desc.get_text(),
        'rate' : rate.get_text(),
        'cate' : list(cate.stripped_strings)
    }

    info.append(data)

for i in info:
    print (i['rate'])
```

-----输出结果-----

### 3.0

那么现在就知道，其实info[]里面总共就一项数据，而且分数刚好等于3，所以按照你大于3分才输出的话，输出为空在逻辑上是正确的。

那么为什么不像视频那样输出结果呢？是因为你的info.append( data )这行代码没有放在for循环里面，也就是只是增加了数据的最后一项，正确的应该是这么写：

```
from bs4 import BeautifulSoup

info = []
with open('/home/ee/Documents/1_2code_of_video/web/new_index.html', 'r') as wb_data:
    Soup = BeautifulSoup(wb_data, "lxml")
    images = Soup.select('body > div.main-content > ul > li > img')
    titles = Soup.select('body > div.main-content > ul > li > div.article-info > h3 > a')
    desc = Soup.select('body > div.main-content > ul > li > div.article-info > p.description')
    rates = Soup.select('body > div.main-content > ul > li > div.rate > span')
    cates = Soup.select('body > div.main-content > ul > li > div.article-info > p.meta-info')

for image,title,desc,rate,cate in zip(images,titles,descs,rates,cates):
    data = {
        'image': image.get('src'),
        'title': title.get_text(),
        'desc' : desc.get_text(),
        'rate' : rate.get_text(),
        'cate' : list(cate.stripped_strings)
    }
    #问题在这里，缩进应该在for循环里面，不然没有添加每一项数据，永远都是最后一项
    info.append(data)

for i in info:
    if float(i['rate'])>3:
        print(i['title'],i['cate'])
```

-----输出结果-----

Sardinia's top 10 beaches ['fun', 'Wow']

How to get tanned ['butt', 'NSFW']

How to be an Aussie beach bum ['sea']

# String 类型使用 split()分割

## 问题描述

在抓取到网页数据之后，字符串规整可以用 `string_obj.split('分隔符')` 方法来分割规整。

如下面抓到的数据是很不规整的：

```
3室
2厅
2卫
- 121 m²
- 21/27层
```

## 问题解决

使用split方法分割，默认是空格分割

```
string = '''
```

```
3室
2厅
2卫
- 121 m²
- 21/27层
```

```
'''
```

```
print (string.split())
```

```
---输出结果----
```

```
['3室', '2厅', '2卫', '-', '121', 'm²', '-', '21/27层']
```

# 文件名跟模块名字冲突错误

## 问题描述

我明明安装了bs4，但为什么总是提示如下信息呢？

```
#!/usr/bin/env python
#-*- coding : utf-8 -*-
#python 3.5

'''Module-function'''

__author__ = 'AJ Kipper'

from bs4 import BeautifulSoup

soup = BeautifulSoup('')
```

错误提示：

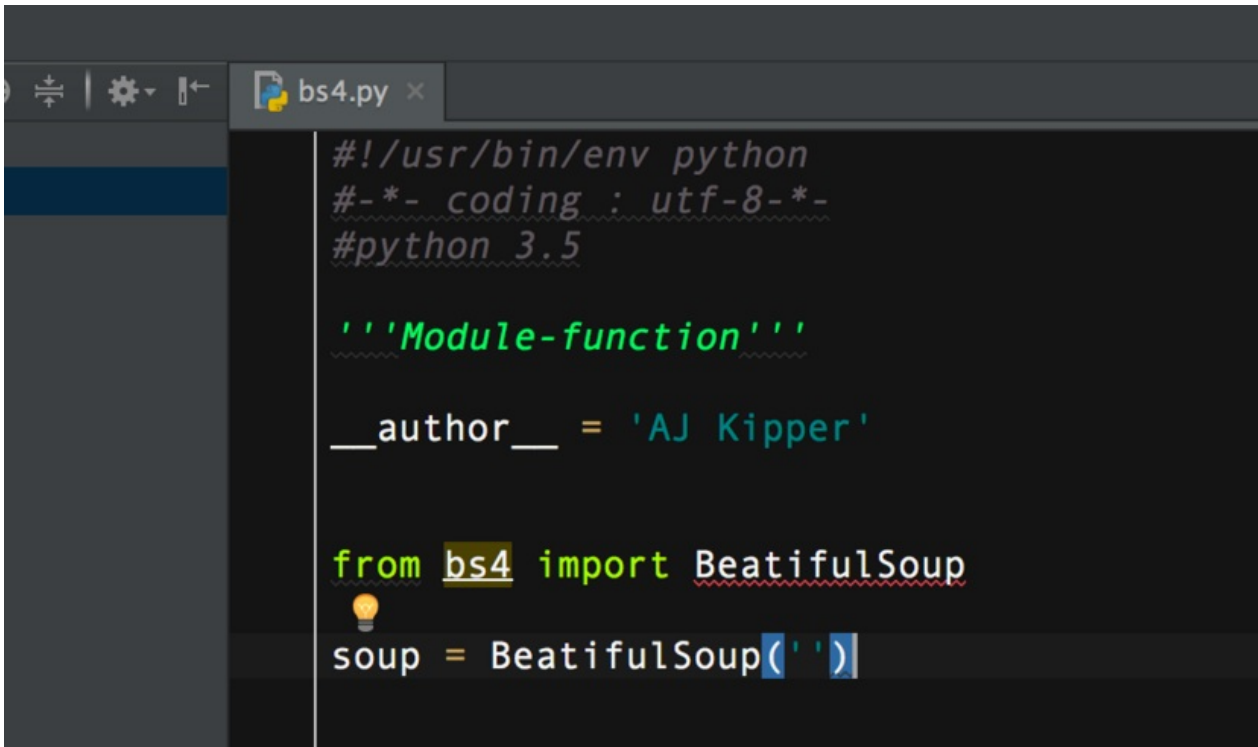
```
/Library/Frameworks/Python.framework/Versions/3.5/bin/python3.5 /Users/Jason/Desktop/test/bs4.py
Traceback (most recent call last):
  File "/Users/Jason/Desktop/test/bs4.py", line 10, in <module>
    from bs4 import BeautifulSoup
  File "/Users/Jason/Desktop/test/bs4.py", line 10, in <module>
    from bs4 import BeautifulSoup
ImportError: cannot import name 'BeautifulSoup'
```

错误提示很奇怪，为什么奇怪呢？因为提示的不是说不能导入bs4，而是提示不能从bs4导入BeautifulSoup，难道我安装的bs4是盗版的？#!=.=

---

## 问题解答

其实原因很简单，看看我的文件名就知道了



```
#!/usr/bin/env python
#-*- coding : utf-8 -*-
#python 3.5

'''Module-function'''

__author__ = 'AJ Kipper'

from bs4 import BeautifulSoup
soup = BeautifulSoup('')
```

我的文件名就是bs4，也就是说，Python模块的导入是按照模块名字先从当前目录寻找，再从包的系统路径里面寻找的，如果我的文件名字就是bs4，那么其实导入的是自身这个文件，没有BeatifulSoup这个类就很正常了，其实不只是自身文件名，只要是自己创建的Python文件，都不能跟Python的库名字一模一样，不然就会出现这种错误。

# with as 的用法

## 问题描述：

Python中with as是怎么用的呢？

在课程视频和源码中，我们有这样的演示：

```
with open('file_path','r+') as file_obj:
    do something...
```

## 问题解答：

其实，打开文件分为最基本的 ①打开文件 -> ②进行文件操作 -> ③关闭文件 这三个步骤，然而在第1步，以读取方式 r 打开文件的话，如果文件本身不存在是会报错的，程序也就停止运行了；读取完数据之后，也要进行文件的关闭，如果打开很多文件都忘记关闭会造成计算机内存的消耗。

所以，Python中除了 with as ，还可以这样打开文件：

```
try:
    f = open('xxx')
    #如果异常
except:
    print 'fail to open'
    exit(-1)
try:
    do something
except:
    do something
finally:
    f.close()
```

也就是 try except finally 用法，不管文件在操作中有没有完成，一定会保证 finally f.close() ，也就是一定会保证文件会关闭。

那么问题来了，Python号称简洁优雅，于是用 with as 的语法特性代替了以上的 try except finally ，只要如以下写：



```
with open('file_path','r+') as file_obj:  
    do something...
```

就不用主动去获取异常和关闭文件，极大简化了编程工作，这就是 `with as` 的基本作用，更深入的可以参考 <http://www.ibm.com/developerworks/cn/opensource/os-cn-pythonwith/>。

# 中文 url 编码问题

## 问题描述:

有些url是带中文后缀的，比如以下这些：

- [http://bj.ganji.com/ershoubijibendiannao/z1/\\_戴尔/](http://bj.ganji.com/ershoubijibendiannao/z1/_戴尔/)
- <http://jixie.huangye88.com/tag/地中衡>

而当你从浏览器复制粘贴下来，或者从页面上抓取下来显示的格式确是这样的：

- [http://bj.ganji.com/ershoubijibendiannao/z1/\\_%E6%88%B4%E5%B0%94/](http://bj.ganji.com/ershoubijibendiannao/z1/_%E6%88%B4%E5%B0%94/)
- <http://jixie.huangye88.com/tag/%E5%9C%B0%E4%B8%AD%E8%A1%A1/>

如果要把第二种格式的编码转换成原来的中文应该怎么做呢？

---

## 问题解答：

其实两种url都是可以正常访问的，没有必要处理，如真是想进行处理的话，可以如以下方式处理：

```
from urllib.parse import unquote
Urls = ['http://bj.ganji.com/ershoubijibendiannao/z1/_%E6%88%B4%E5%B0%94/', 'http://jixie.huangye88.com/tag/%E5%9C%B0%E4%B8%AD%E8%A1%A1/']

for url in Urls:
    print(unquote(url, encoding="utf-8"))

+++++++
输出结果
+++++++
http://bj.ganji.com/ershoubijibendiannao/z1/_戴尔/
http://jixie.huangye88.com/tag/地中衡/
```

## 内置函数 `zip()` 的使用

### 翻译中文文档

#### `zip([iterable, ...])`

该函数返回一个元组的列表，其中第  $i$  个元组包含每个参数序列的第  $i$  个元素。返回的列表长度被截断为最短的参数序列的长度。当多个参数都具有相同的长度时，`zip()` 类似于带有一个初始参数为 `None` 的 `map()`。只有一个序列参数时，它返回一个1元组的列表。没有参数时，它返回一个空的列表。

可以保证迭代按从左向右的计算顺序。这使得使用 `zip(\*[iter(s)]*n)` 来将一系列数据分类归并为长度为  $n$  的组成为习惯用法。

`zip()` 与 `*` 操作符一起可以用来 `unzip` 一个列表：

```
>>> x = [1, 2, 3]
>>> y = [4, 5, 6]
>>> zipped = zip(x, y)
>>> zipped
[(1, 4), (2, 5), (3, 6)]
>>> x2, y2 = zip(*zipped)
>>> x == list(x2) and y == list(y2)
True
```

以前，`zip()` 要求至少一个参数，且 `zip()` 抛出一个 `TypeError` 异常，而不是一个空的列表。

---

官方英文文档引用：

## zip(\*iterables)

Make an iterator that aggregates elements from each of the iterables.

Returns an iterator of tuples, where the i-th tuple contains the i-th element from each of the argument sequences or iterables. The iterator stops when the shortest input iterable is exhausted. With a single iterable argument, it returns an iterator of 1-tuples. With no arguments, it returns an empty iterator. Equivalent to:

```
def zip(*iterables):
    # zip('ABCD', 'xy') --> Ax By
    sentinel = object()
    iterators = [iter(it) for it in iterables]
    while iterators:
        result = []
        for it in iterators:
            elem = next(it, sentinel)
            if elem is sentinel:
                return
            result.append(elem)
        yield tuple(result)
```

The left-to-right evaluation order of the iterables is guaranteed. This makes possible an idiom for clustering a data series into n-length groups using `zip([iter(s)]n)`. This repeats the same iterator n times so that each output tuple has the result of n calls to the iterator. This has the effect of dividing the input into n-length chunks.

`zip()` should only be used with unequal length inputs when you don't care about trailing, unmatched values from the longer iterables. If those values are important, use `itertools.zip_longest()` instead.

`zip()` in conjunction with the `*` operator can be used to unzip a list:

```
>> x = [1, 2, 3]
>> y = [4, 5, 6]
>> zipped = zip(x, y)
>> list(zipped)
[(1, 4), (2, 5), (3, 6)]
>> x2, y2 = zip(*zip(x, y))
>> x == list(x2) and y == list(y2)
```

# Python 读写 CSV文件

Python中读写CSV文件，可以用到csv这个库，这个库是Python的标准自带库，不用第三方下载安装，直接使用就行。

简单的读写使用可以参考这个链接

<https://github.com/AJKipper/PythonLibrarys/tree/master/csv>

---

参考文档：

- [官方英文文档](#)
- [中文文档](#)

## 写路径问题

初学者在学习时，往往会遇到一个错误，如下：

```
File "E:/python/test.py", line 77
    path = "C:\Users\Administrator\Desktop"
          ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3:
truncated \UXXXXXXXX escape
```

这个错误其实是因为这行代码引起的：

```
path = "C:\Users\Administrator\Desktop"
```

在这行代码中，有几个"\"，而"\"在python中表示转义，转义是什么意思呢？举个例子：执行以下代码将打印一个换行：

```
print("\n")
```

```
>>> print("12")
12
>>> print("1\n2")
1
2
>>> |
```

为什么不是输出"\n"，而是输出一个换行呢，因为"n"经过"\"转义之后，成为一个换行符。好的，现在回到我们的问题上，我们的变量path中的值带有"\"，而"\"打算将U转义为有意义的符号，而"\"并未能如愿，因为没有对应的转义字符。这时只能报上述的错误SyntaxError: (unicode error) 'unicodeescape'。

## 那么该如何解决呢？

第一种办法：

在字符串前加一个字母r，如下：

```
path=r"C:\Users\Administrator\Desktop"
```

在字符串前加个r是为了告诉编译器这个string是个raw string，不要转义。

第二种办法：

将字符串中的反斜杠换成正斜杠，如下：

path="C:/Users/Administrator/Desktop"

---

关于路径问题，在不同系统中，写法都大同小异。而这些问题无外乎源于这两方面：正反斜杠，编码（无法识别中文等）。

# python简单的排错技巧

很多初学者运行程序后，得不到自己的预期的结果，或者得到更糟的结果：程序在中途就停止运行。

遇到这种问题我们除了凭借自己平时积累的经验来排错之外，更多的是靠我们一步一步地调试，直至程序运行成功。

不管是初学者，还是经验老道的程序员，在调试时都不会离开**print()**-打印这个方法，**print()**方法在程序投入到生产环境时更多是以这样一种形式存在：

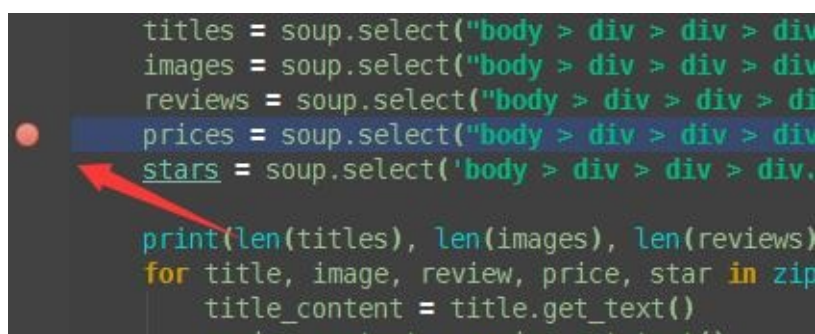
```
#print(test)
```

**print()**被注释掉了，为什么呢，因为当程序代码在被编写时，**print()**可以帮助程序员查看程序运行的结果。

所以当你对自己程序的结果产生疑惑又不能一下子找到原因时，这时你可以从上往下(或者从下往上)一步一步**print**过程中的某个变量，这时往往就能找到问题所在。

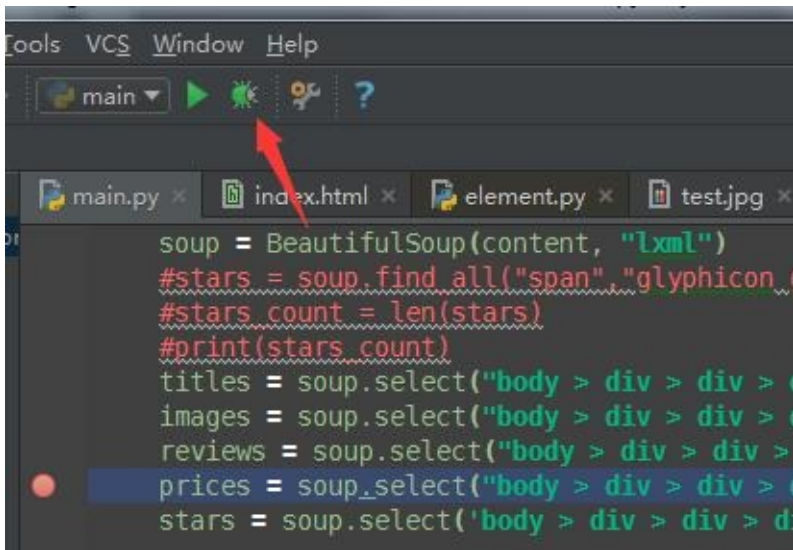
平时使用的**pycharm**自身自己携带了很多功能，包括基本代码的缩进以及语法检查。而其中也包括断点调试功能：

这个调试会稍微复杂点，首先我可以设置一个断点：

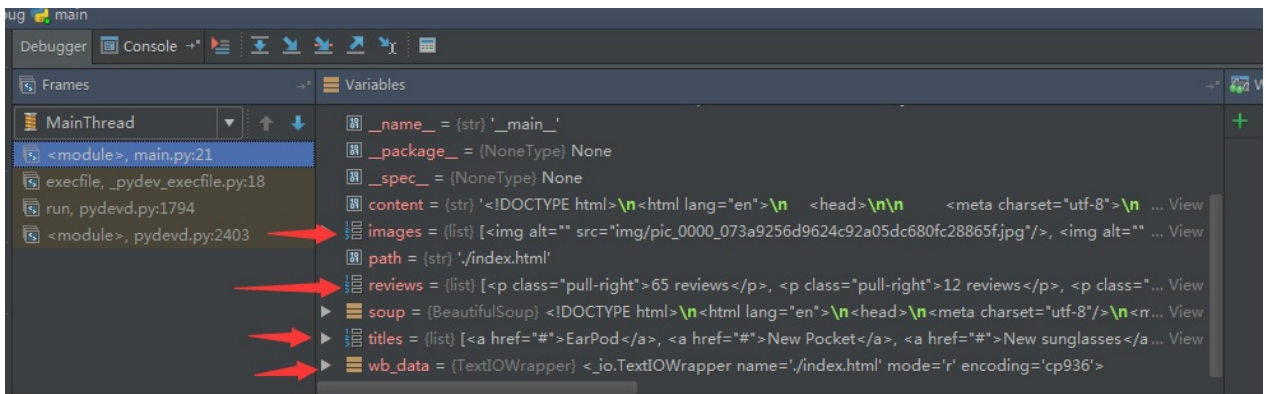


再点击我们的Debug:



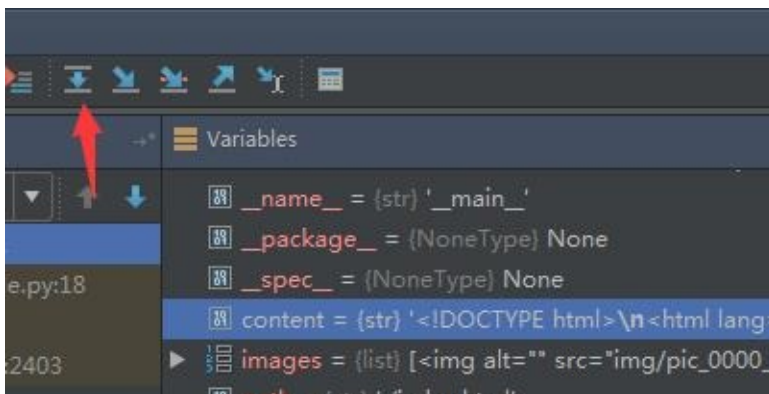


然后再看看底下：



此时就可以看到我们程序运行时的变量啦。

还可以向下一步一步执行：



具体如何使用Debug网上有很多详细的教程，在此只是抛砖引玉。

# PyCharm 编辑器问题集

## 内容概括

PyCharm是一种Python IDE，带有一整套可以帮助用户在使用Python语言开发时提高其效率的工具，比如调试、语法高亮、Project管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，该IDE提供了一些高级功能，以用于支持Django框架下的专业Web开发。

所以，PyCharm是一个强大的编辑器，使用起来也会遇到很多问题，问题集里面包括了插件安装问题，版本选择问题，具体使用问题等。

---

## 参考资料

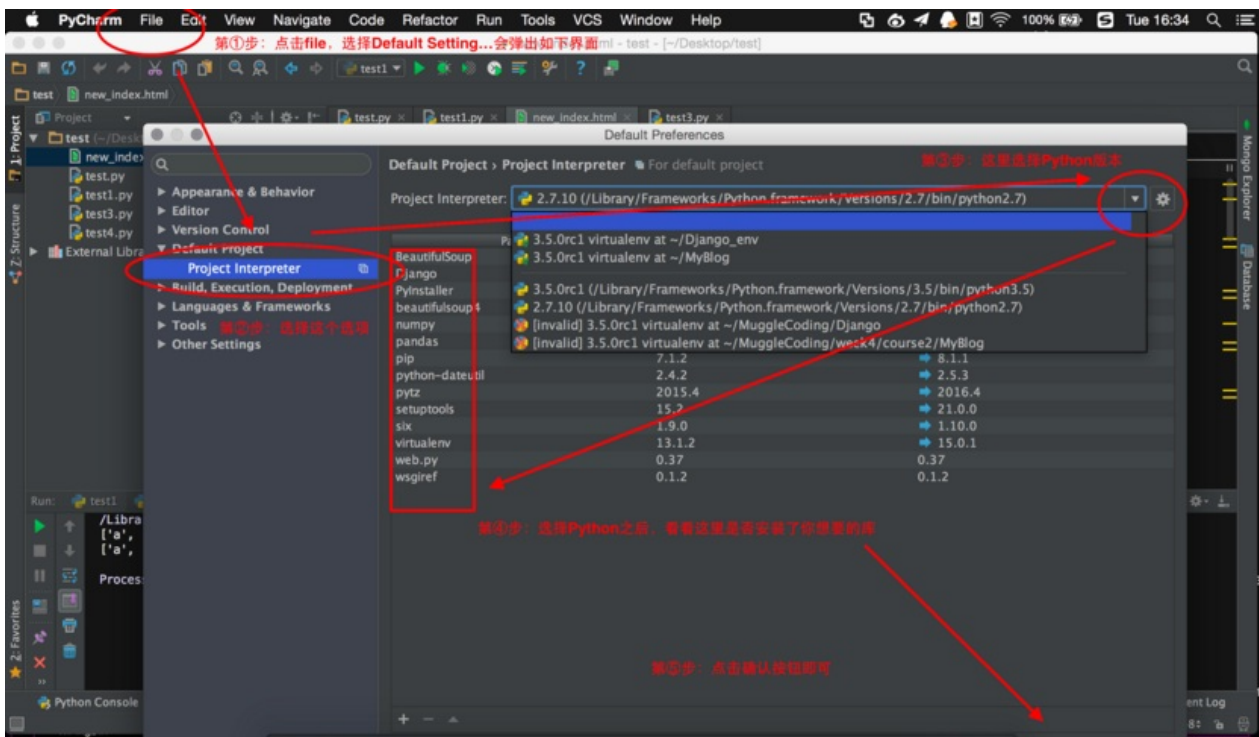
1. [PyCharm官网](#)

## 安装库之后，运行提示没有库

在使用PyCharm编辑器运行代码的时候，有时候会提示没有安装某个库，但是自己却明明安装了这个库。这样的错误原因有很多，下面总结一下有可能的情况。

### 情况一

已经安装了某个库的Python并不是你在PyCharm编辑器默认运行的Python。所以要更换默认Python，这个可以进行如下的操作：



### 情况二

如果你按照上面的情况设置好了还是出错，那就看看自己创建文件的名称是不是跟库的名称一模一样冲突了，自己创建的Python文件不能跟Python的库名称一样。

如果这两种情况都设置好了还出问题，请向老师提问。



## 第一周课程问题集

# BeautifulSoup4问题集

---

## 内容概括

BeautifulSoup4是一个可以从HTML或XML文件中提取数据的Python库.它能够通过你喜欢的转换器实现惯用的文档导航,查找,修改文档的方式。在爬虫程序中，它更是不可或缺的神器。

由于BeautifulSoup也是一个比较复杂的库，所以熟练的掌握也需要一定的时间，问题集也是收集了一般会遇到的问题，更深入的研究可以参考官方文档。

---

## 参考资料

1. [Beautiful Soup 4.2.0 中文文档](#)
2. [Beautiful Soup 4.4.0 documentation](#)

## 错误提示锦集

## soup.select ( )选择标签问题

这个知识点的内容在第一周第二节课程里面有详细的解说，弄不懂的同学可以多看几遍。

代码原型是这样的：

```
import requests
from bs4 import BeautifulSoup

web_data = requests.get('url')
soup = BeautifulSoup(web_data.text, 'lxml')
data = soup.select('选择内容')
print (data)
```

一个典型的例子

关于'选择内容'方面还是有同学有疑问。

比如这个url：<http://www.meihua.info/?page=1>

爆文效应明显，19%的10万+贡献了72%的阅读数

500强4月推送的内容中有明确转载来源的内容有3262篇，占0.3%，这些转载的平均阅读数为6万，比500强整体单篇平均阅读数高1万。转载方面，500强发布的内容中有转载篇数小，转载来源多样化的特征。

今天 ·  阅读 65 ·  业界 ·  微信，阅读量

要截取一篇文章的以下字段：

- 日期（今天）
- 阅读量（阅读65）
- 类型（业界）
- 文章标签（微信，阅读量）

那么在浏览器中鼠标右键点击审查 -> 找到字段的具体代码 -> 右键选择Copy -> Copy selector ,如下：

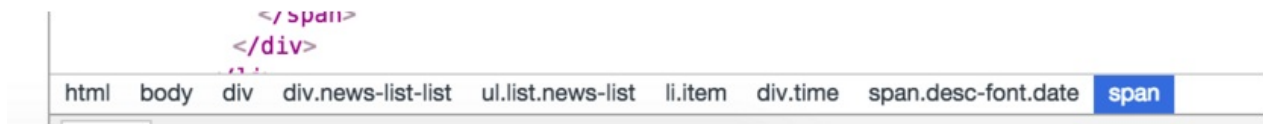
复制的内容如下：



```
body > div.main-frame > div.news-list-list > ul > li:nth-child(1) > div.time > span >
span:nth-child(1)
```

如果直接把这个作为`select ( )`的参数是不行的，`li:nth-child(1)`要去掉，这是CSS的高级选择器，而且`select ( )`不支持这种类型，去掉就行。然后还要查看html结构的层级，一步步加上去，不懂的可以多尝试多总结。

### html层级结构



以下是我修改的`select ( )`参数内容：

```
body > div.main-frame > div.news-list-list > ul.list.news-list > li > div.time > span.
desc-font.date > span
```

运行代码试一下：

```
import requests
from bs4 import BeautifulSoup as bs

web_data = requests.get('http://www.meihua.info/?page=1')

soup = bs(web_data.text, 'lxml')
post_date = soup.select('body > div.main-frame > div.news-list-list > ul.list.news-list > li > div.time > span.desc-font.date > span')[0]

print (post_date.get_text())

--输出结果--
今天
```

其它字段可以自行尝试。

# 编码输出问题

## 问题描述

有个同学在抓取这个网页 <http://www.ikanchai.com/> 的时候，出现乱码，如下：

```
import requests
from bs4 import BeautifulSoup as bs

web_data = requests.get('http://www.ikanchai.com/')
soup = bs(web_data.text, 'lxml')
dates = soup.select('div.sort.channel.clearfix > ul > li > a')

for data in dates:
    print (data)
-----输出结果-----
<a href="http://www.ikanchai.com/view/">è$,ç,¹</a>
<a href="http://www.ikanchai.com/start/">å^>æ$•</a>
<a href="http://www.ikanchai.com/evaluation/">è¯¸,æµ</a>
<a href="http://www.ikanchai.com/vr/">VR</a>
<a href="http://www.ikanchai.com/push/">ä, "æ </a>
<a href="http://app.ikanchai.com/roll.php">å$¨æ€</a>
```

那么其实这是编码问题，加一句 `web_data.encoding = 'utf-8'` 就可以了，具体的编码选择在html文档开头有说明。

## 问题解决

更正之后的代码：

```
import requests
from bs4 import BeautifulSoup as bs

web_data = requests.get('http://www.ikanchai.com/')
web_data.encoding = 'utf-8'
soup = bs(web_data.text, 'lxml')
dates = soup.select('div.sort.channel.clearfix > ul > li > a')

for data in dates:
    print (data)

-----输出结果-----
<a href="http://www.ikanchai.com/view/">观点</a>
<a href="http://www.ikanchai.com/start/">创投</a>
<a href="http://www.ikanchai.com/evaluation/">评测</a>
<a href="http://www.ikanchai.com/vr/">VR</a>
<a href="http://www.ikanchai.com/push/">专栏</a>
<a href="http://app.ikanchai.com/roll.php">动态</a>
```

## get\_text 和 get()的使用

- `get_text()`，是返回选择的标签文本，具体可以参考文档：[http://beautifulsoup.readthedocs.io/zh\\_CN/latest/#get-text](http://beautifulsoup.readthedocs.io/zh_CN/latest/#get-text)

- 
- `get('')`，这是选择标签中的属性(也就是里面有=符号的左边)，比如

```
<li><a class="reference internal" href="#next-siblings-previous-siblings">
```

中，选择 `li > a` 这个标签之后，则可以用 `get('href')` 获取其中的链接。

## find\_all()和 select()的区别

在soup中，我们通常都是使用lxml去解析我们得到的源码：

```
soup = BeautifulSoup(wb_data.text, "lxml")
```

之后我们通常使用select()或find\_all() 去得到我们想要的节点。那么什么时候用select，又是什么时候用find\_all呢？

select()是使用CSS选择器的语法找到tag 如：

```
soup.select("title")
# [<title>The Dormouse's story</title>]

通过tag标签逐层查找：
soup.select("body a") #body子孙标签中的a标签
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

找到某个tag标签下的直接子标签
soup.select("p > a:nth-of-type(2)")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

find\_all()方法搜索当前tag的所有tag子孙节点,并判断是否符合过滤器的条件

```
soup.find_all("title")
# [<title>The Dormouse's story</title>]

css_soup.find_all("p", class_="body")
# [<p class="body strikeout"></p>]
```

select()和find\_all()，我们平时的爬虫程序，哪个方便就用哪个吧。具体可看官方文档：

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/index.zh.html>

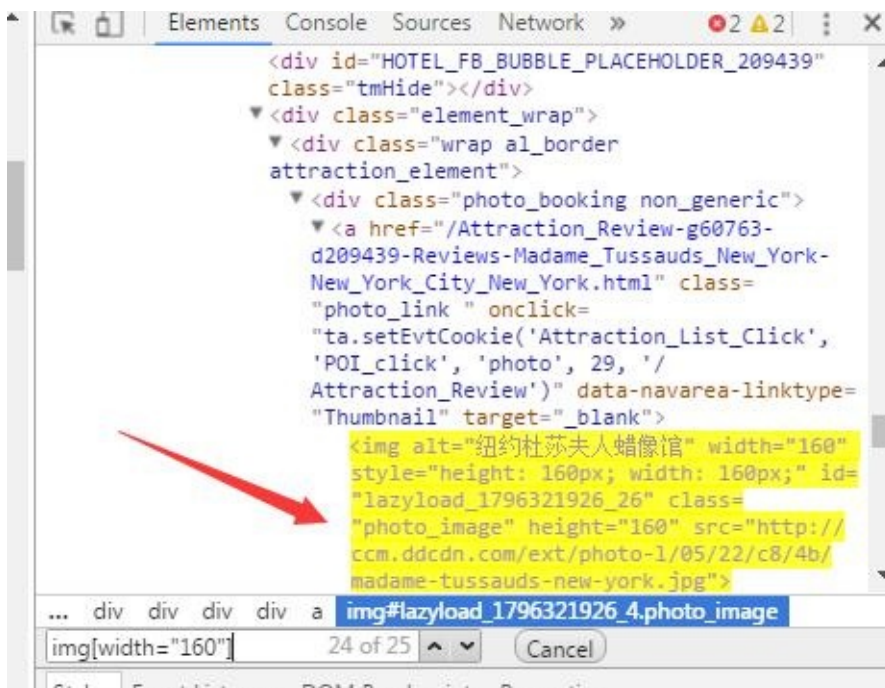
## soup查找tag，结果为空[]

### 问题描述：

为什么soup.select("img[width='160']")结果是[]？网址：

[http://www.tripadvisor.cn/Attractions-g60763-Activities-oa30-New\\_York\\_City\\_New\\_York.html#ATTRACTION\\_LIST](http://www.tripadvisor.cn/Attractions-g60763-Activities-oa30-New_York_City_New_York.html#ATTRACTION_LIST)

而我在开发者工具中，明明是有的呀？如：



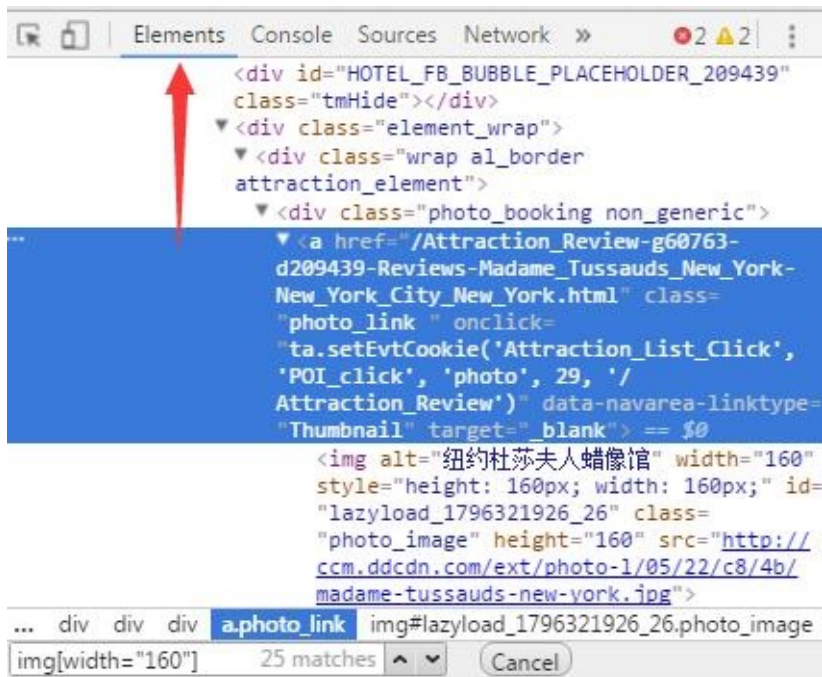
### 问题解决

首先你需要排除几个影响因素：

- 1、selector是否写对？
- 2、网页源码中是否含有我要找的内容？
- 3、在headers中加入Agent-User或者cookie是否有用？
- 4、伪装成手机浏览器是否有用？
- 5、加入Referer是否有用？

1、selector是否写对？

F12 打开我们的开发者工具，在Elements中Ctrl+F 把我们的selector复制进去，有找到我们想要的内容就说明selector是没错的，转问题2，否则，改一改selector。



## 2、网页源码中是否含有我要找的内容？

将我们想要的内容复制，如：



madame-tussauds-new-york.jpg

在浏览器网页上鼠标右键 查看网页源码，查找：

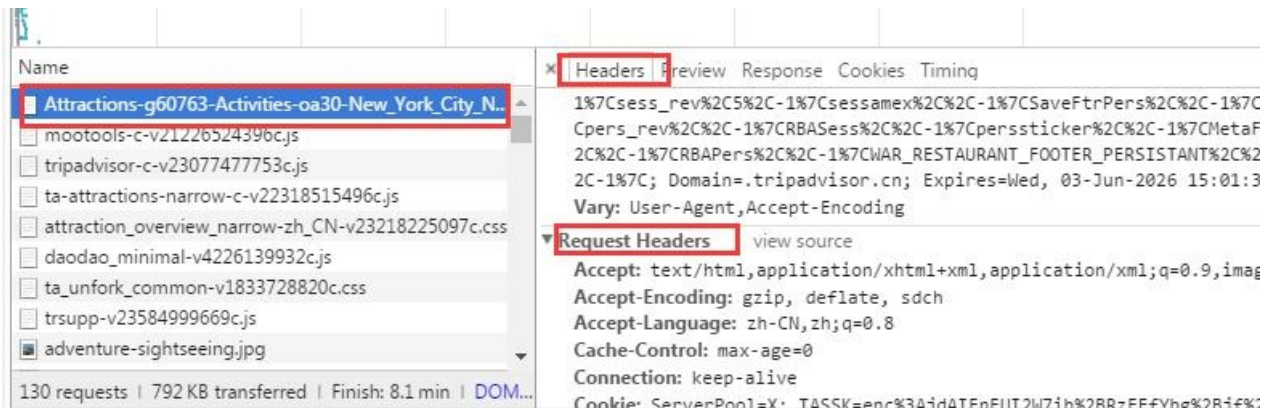
```
1/08/59/hard-hat-tour-corridor.jpg", "scroll": false, "tagType": "img", "id": "la:  
2/dd/5d/the-manhattan-bridge.jpg", "scroll": false, "tagType": "img", "id": "la:  
1/d8/e6/the-church.jpg", "scroll": false, "tagType": "img", "id": "lazyload_904  
a/81/3f/view-from-5th-avenue.jpg", "scroll": false, "tagType": "img", "id": "la:  
:/16/44/upper-west-side.jpg", "scroll": false, "tagType": "img", "id": "lazyloa  
2/c8/4b/madame-tussauds-new-york.jpg", "scroll": false, "tagType": "img", "id":  
1/c5/15/the-new-amsterdam-theater.jpg", "scroll": false, "tagType": "img", "id"
```

在源码中找到了我们想要的内容，但是却出现在了script中，我们没办法使用lxml正常解析。为什么没有出现在该出现的位置呢？因为这个数据是通过js脚本加到我们网页的主体内容上的。因为我们的程序是没有运行js脚本的，所以没办法使用正常的方法(select)获取。

尝试下问题3

### 3、在headers中加入Agent-User或者cookie是否有用？

还是用我们的开发者工具，在network一栏中，找到Agent-User和cookie: 选择第一个查看requests headers：



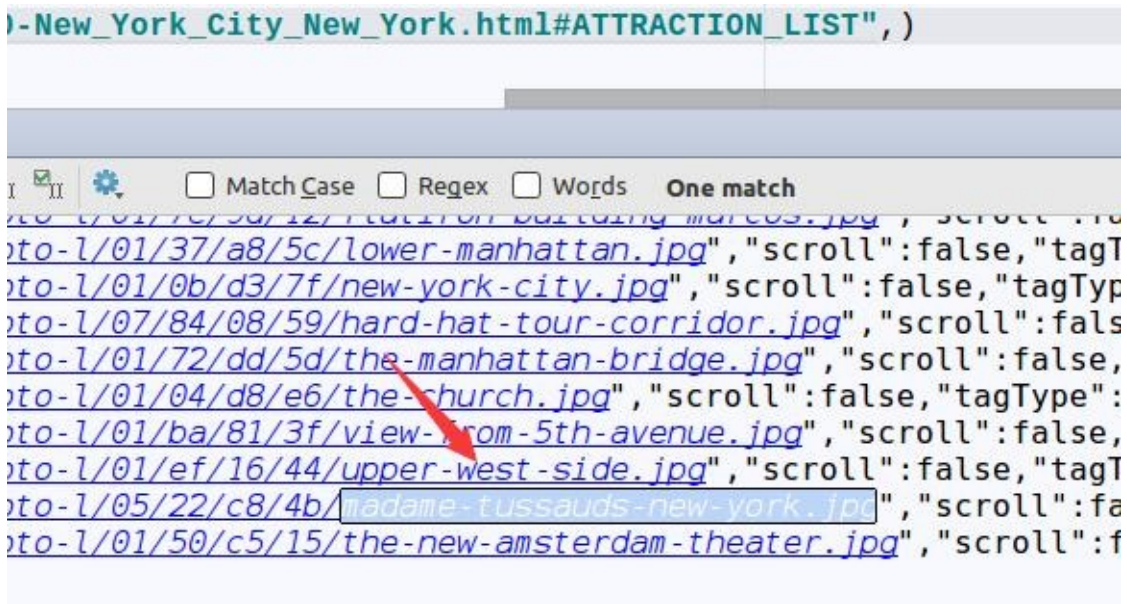
这时我们就要用程序去尝试会更快，写好程序：



```
#coding:utf-8
import requests
from bs4 import BeautifulSoup

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36',
    'Cookie': 'ServerPool=X; TASSK=enc%3AjdAIEnEUI2W7ib%2BRzEEfYbg%2BjF%2FGJ2FeuxWecNuh1b9%2BG4uSJWL14%2BMwdNe1wynX; TAUnique=%1%enc%3AyptiXtNF4o6wsKQVc450sHwVy9f4akSHVUbAcqKV9Fo2jHwltRJPgQ%3D%3D; TAPD=tripadvisor.cn; __gads=ID=f756d423c624c3a6:T=1465124473:S=ALNI_MYb2KIun9_Lx47a9fojaf8YSf05-Q; ki_t=1465124476104%3B1465124476104%3B1465124476104%3B1%3B1; ki_r=; MobileLastViewedList=%1%2FAttractions-g60763-Activities-oa30-New_York_City_New_York.html; taMobileRV=%1%7B%2210028%22%3A%5B60763%5D%7D; a5281_times=1; BD_vm133w=_; BD_vm133w=_; TATravelInfo=V2*A.2*MG.-1*HP.2*FL.3; roybatty=Aph3S5oN23jNlA08aqSjGGadXhY2Yxug5ybfhzb2XdppH0Y5EPoFXpmj4Vb3BBS322SG1lbpTddueH2Z951T0U1%2BAYR%2Fc4TKhu%2FtWM7x1%2BSg7mbdFgz0XtLQMFe5M4q4NsD%2BUDI9CYtjwQ50Ixpt8o8SBpRMEkcf45kII%2FcRRup%2C1; Hm_lvt_2947ca2c006be346c7a024ce1ad9c24a=1465124800; Hm_lpvt_2947ca2c006be346c7a024ce1ad9c24a=1465135673; a2117_pages=4; a2117_times=2; Hm_lvt_69d068dda1da018d9b4ba380c7b92ee3=1465124473; Hm_lpvt_69d068dda1da018d9b4ba380c7b92ee3=1465135673; NPID=; TASession=%1%V2ID.B273D66C4BEC744CE37CDD3CA404F787*SQ.14*LS.Attractions*GR.46*TCPAR.88*TBR.50*EXEX.38*ABTR.30*PPRP.51*PHTB.78*FS.10*CPU.76*HS.popularity*ES.popularity*AS.popularity*DS.5*SAS.popularity*FPS.oldFirst*FA.1*DF.0*LP.%2FLangRedirect%3Fauto%3D3%26origin%3Dzh%26pool%3DX%26returnTo%3D%252FAttractions-g60763-Activities-oa30-New_York_City_New_York%5C.html*IR.3*OD.zh*RT.0*FLO.60763*TRA.true*LD.60763; CM=%1%HanaPersist%2C%2C-1%7Ct4b-pc%2C%2C-1%7CHanaSession%2C%2C-1%7CFtrSess%2C%2C-1%7CRCPers%2C%2C-1%7CHomeAPers%2C%2C-1%7CWShadeSeen%2C%2C-1%7CRCsess%2C%2C-1%7CFtrPers%2C%2C-1%7CHomeASess%2C%2C-1%7Csh%2C%2C-1%7Cpsamex%2C%2C-1%7C2016sticksess%2C%2C-1%7CCCPers%2C%2C-1%7CCCSess%2C%2C-1%7CWAR_RESTAURANT_FOOTER_SESSION%2C%2C-1%7Cb2bmcsess%2C%2C-1%7Csesssticker%2C%2C-1%7C%24%2C%2C-1%7C2016stickpers%2C%2C-1%7Ct4b-sc%2C%2C-1%7CMC_IB_UPSELL_IB_LOGOS2%2C%2C-1%7Cb2bmcpers%2C%2C-1%7CMC_IB_UPSELL_IB_LOGOS%2C%2C-1%7Csess_rev%2C4%2C-1%7Csessamex%2C%2C-1%7CSaveFtrPers%2C%2C-1%7CSaveFtrSess%2C%2C-1%7Cpers_rev%2C%2C-1%7CRBASess%2C%2C-1%7Cperssticker%2C%2C-1%7CMetaFtrSess%2C%2C-1%7Ccmds%2C%2C-1%7CRBAPers%2C%2C-1%7CWAR_RESTAURANT_FOOTER_PERSISTANT%2C%2C-1%7CMetaFtrPers%2C%2C-1%7C; TAUD=LA-1465124469834-1*LG-13046557-2.0.F*LD-13046559-....; TAReturnTo=%1%2FAttractions-g60763-Activities-oa30-New_York_City_New_York.html',
}
resp = requests.get("http://www.tripadvisor.cn/Attractions-g60763-Activities-oa30-New_York_City_New_York.html#ATTRACTION_LIST", headers=headers)
soup = BeautifulSoup(resp.text, "lxml")
print(soup)
```

在headers 中加入User-Agent、cookie等。在得到的结果中，我们依然只是在script中找到我们想要的内容(可以将网页源码写入文件中，方便查看)。



尝试伪装成手机浏览器：

#### 4、伪装成手机浏览器是否有用？

修改一下代码，把User-Agent改一下，改成iphone。怎么才会有iphone的User-Agent呢？课程视频中有详细解说，再此不再赘述。

```

headers = {
    'User-Agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 9_1 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13B143 Safari/601.1',
    'Cookie': 'ServerPool=X; TASSK=enc%3AjdAIEnEUI2W7ib%2BRzEEfYbg%2Bjf%2FGJ2FeuxWecNuh1b9%2BG4uSJWL14%2BMwdNe1wynX; TAUnique=%1%enc%3AyptiXtNF4o6wsKQVc450sHwVy9f4akSHVUBAcqKV9Fo2jHwltRJPQG%3D%3D; TAPD=tripadvisor.cn; __gads=ID=f756d423c624c3a6:T=1465124473:S=ALNI_MYb2KIun9_Lx47a9fojaf8YSf05-Q; ki_t=1465124476104%3B1465124476104%3B1465124476104%3B1%3B1; ki_r=; MobileLastViewedList=%1%2FAttractions-g60763-Activities-oa30-New_York_City_New_York.html; taMobileRV=%1%7B%2210028%22%3A%5B60763%5D%7D; a5281_times=1; BD_vm133w=_; BD_vm133w=_; TATravelInfo=V2*A.2*MG.-1*HP.2*FL.3; roybatty=Aph3S5oN23jNlA08aqSjGGadXhY2Yxug5ybfhzb2XdppH0Y5EPoFXpmj4Vb3BBS322SG1lbpTddueH2Z951T0U1%2BAYR%2Fc4TKhu%2FtWM7x1%2BSg7mbdFgz0XTLQMFfe5M4q4NSd%2BUDI9CYtjwQ50Ixpt8o8SBpRMEkcf45kII%2FcRRup%2C1; Hm_lvt_2947ca2c006be346c7a024ce1ad9c24a=1465124800; Hm_lpv_2947ca2c006be346c7a024ce1ad9c24a=1465135673; a2117_pages=4; a2117_times=2; Hm_lvt_69d068dda1da018d9b4ba380c7b92ee3=1465124473; Hm_lpv_69d068dda1da018d9b4ba380c7b92ee3=1465135673; NPID=; TASession=%1%V2ID.B273D66C4BEC744CE37CDD3CA404F787*SQ.14*LS.Attractions*GR.46*TCPAR.88*TBR.50*EXEX.38*ABTR.30*PPRP.51*PHTB.78*FS.10*CPU.76*HS.popularity*ES.popularity*AS.popularity*DS.5*SAS.popularity*FPS.oldFirst*FA.1*DF.0*LP.%2FLangRedirect%3Fauto%3D%26origin%3Dzh%26pool%3DX%26returnTo%3D%252FAttractions-g60763-Activities-oa30-New_York_City_New_York%5C.html*IR.3*OD.zh*RT.0*FLO.60763*TRA.true*LD.60763; CM=%1%HanaPersist%2C%2C-1%7Ct4b-pc%2C%2C-1%7CHanaSession%2C%2C-1%7CFtrSess%2C%2C-1%7CRCPers%2C%2C-1%7CHomeAPers%2C%2C-1%7CWSHadeSeen%2C%2C-1%7CRCsess%2C%2C-1%7CFtrPers%2C%2C-1%7CHomeASess%2C%2C-1%7Csh%2C%2C-1%7Cpssamex%2C%2C-1%7C2016sticksess%2C%2C-1%7CCCPers%2C%2C-1%7CCCSess%2C%2C-1%7CWAR_RESTAURANT_FOOTER_SESSION%2C%2C-1%7Cb2bmcsess%2C%2C-1%7Csesssticker%2C%2C-1%7C%24%2C%2C-1%7C2016stickpers%2C%2C-1%7Ct4b-sc%2C%2C-1%7CMC_IB_UPSELL_IB_LOGOS2%2C%2C-1%7Cb2bmcpers%2C%2C-1%7CMC_IB_UPSELL_IB_LOGOS2%2C%2C-1%7Csess_rev%2C4%2C-1%7Csessamex%2C%2C-1%7CSaveFtrPers%2C%2C-1%7CSaveFtrSess%2C%2C-1%7Cpers_rev%2C%2C-1%7CRBASess%2C%2C-1%7Cperssticker%2C%2C-1%7CMetaFtrSess%2C%2C-1%7Ccmds%2C%2C-1%7CRBAPers%2C%2C-1%7CWAR_RESTAURANT_FOOTER_PERSISTANT%2C%2C-1%7CMetaFtrPers%2C%2C-1%7C; TAUD=LA-1465124469834-1*LG-13046557-2.0.F*LD-13046559-....; TAReturnTo=%1%2FAttractions-g60763-Activities-oa30-New_York_City_New_York.html'
}

resp = requests.get("http://www.tripadvisor.cn/Attractions-g60763-Activities-oa30-New_York_City_New_York.html#ATTRACTION_LIST", headers=headers)
soup = BeautifulSoup(resp.text, "lxml")
print(soup)

```

在次查找：



找到了，那我们现在就使用我们调试得到的User-Agent来获取我们想要的内容。

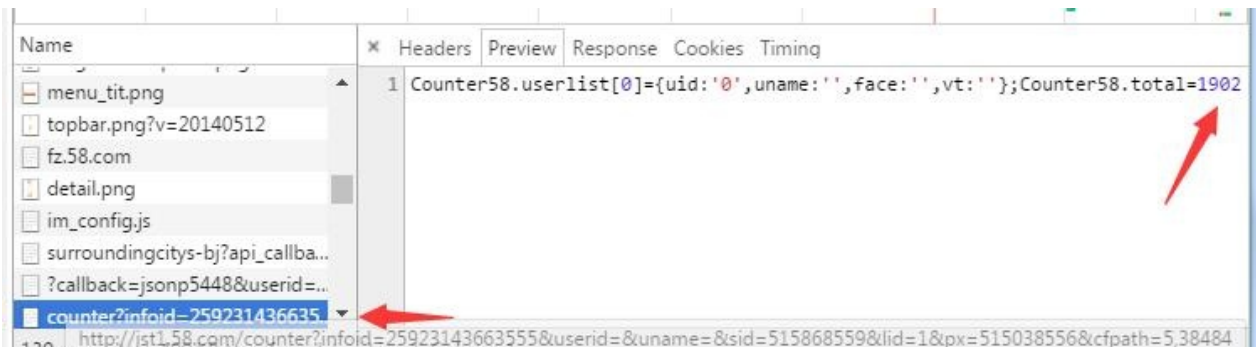
## 5、加入Referer是否有用？

soup查找tag，结果为空[]

如：



我们要抓取这个访问量，在网页源码中没找到这个访问量，但是在network中找到了一个api接口：

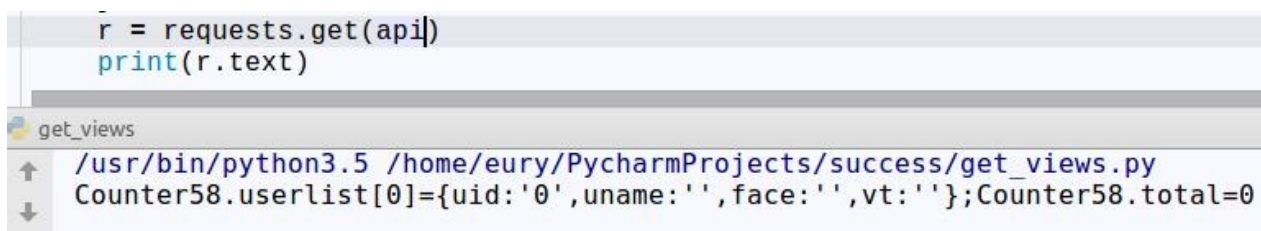


<http://jst1.58.com/counter?>

[infoid=25923143663555&userid=&uname=&sid=515868559&lid=1&px=515038556&cfpath=5,38484](http://jst1.58.com/counter?infoid=25923143663555&userid=&uname=&sid=515868559&lid=1&px=515038556&cfpath=5,38484)

简单分析网址，infoid是我们爬取链接的网页id 在这以为已经成功拿到访问量， 尝试使用程序获取访问量：

得到的结果却是：0



尝试加入Referer。 还是利用类似问题3和问题4的方法，拿到Referer，在headers加入Referer。

```
def get_views_from(url):
    url_path = url.split("?")[0]
    url_last_part = url_path.split('/')[-1]
    info_id = url_last_part.strip('x.shtml')
    api = 'http://jst1.58.com/counter?infoid={}'.format(info_id)
    headers = {
        'User-Agent': r'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36',
        'Referer': r'http://bj.58.com/pingbandiannao/{}x.shtml'.format(info_id),
    }
    r = requests.get(api, headers=headers)
    print(r.text)
```

get\_views

```
/usr/bin/python3.5 /home/eury/PycharmProjects/success/get_views.py
Counter58.userlist[0]={uid:'0',uname:'',face:'',vt:''};Counter58.total=1152
```

现在已经可以看到我们的访问量啦~

总结一下：在无法正确拿到数据时，请考虑以下因素：

- 1、selector是否写对？
- 2、网页源码中是否含有我要找的内容？
- 3、在headers中加入Agent-User或者cookie是否有用？
- 4、伪装成手机浏览器是否有用？
- 5、加入Referer是否有用？
- 6、其他

## soup.select()尽量不使用完整selector

使用beautifulsoup来定位我们想要的信息，通常步骤是：

谷歌浏览器 -> 审查元素 -> *copy selector*

这时我们通常会得到一个完整的路径：

```
Main > div:nth-child(2) > div:nth-child(18) > table > tbody > tr > td:nth-child(3) > span.item_title > a
```

然后把这个selector放入select去定位我们想要的数据。

而我们常常遇到一种我们并不想要的结果：[]

一个空列表，因为我们select没准确定位我们想要的的数据，所以得到一个空列表，难道是我们的select错了吗？

并不是的，而是因为我们复制回来的selector是浏览器上的selector，我们平时在浏览器上看到的都是经过js脚本的加工，所以selector也是经过加工的。

而我们程序爬取到的网页并没有经过浏览器加工，所以所需的selector有可能和浏览器上的不一样，也就是为什么我们会得到一个空列表。

## 解决办法

尽量使用较短的selector去定位我们的数据，一般复制回来的selector前半部分可以不要，如果还是没办法准确定位，那再往前加。

## pip 问题集

# pip 怎么更新 Python 包？

## pip 更新 Python 包

pip 是一个可执行的脚本文件，设置了环境变量就可以在终端(CMD)里面使用 pip 命令，而这个命令有一个 `--upgrade` 或者 `-U` 参数，英文的意思是 升级，用来更新包。

所以命令如下：

- `pip install --upgrade <包的名字>`
- `pip install -U <包的名字>`

---

## pip 自我更新

由于 pip 本身就是一个 Python 的包，那么用 pip 来自我更新也是可以的，命令如下：

- `pip install --upgrade pip`
- `pip install -U pip`

或者用 Python 命令参数 `-m` 选项来安装也是可以的，`-m` 的意思是用 Python 解释器来运行 pip 再更新，Python 更新 pip 命令如下：

- `python -m pip install --upgrade pip`
- `python -m pip install -U pip`



## 代理ip问题集

## 如何获取代理ip

很多同学在爬取数据时，常常要加一行代码：

```
time.sleep(2)
```

加这行程序是为了防止我们的爬虫程序对我们要爬取目标网址的访问频率过快，导致目标网站误以为我们是攻击行为，进而被封ip,而不能继续访问该网站。所以要加以上那行程序。

为了防止我们被目标网站封掉ip，并且加快我们爬取的速度，聪明的你一定会想到使用代理，利用多个ip对我们的目标网站进行快速地爬取。

那么我们该如何获取代理ip呢：使用谷歌或者百度 代理ip,会出现很多提供代理ip的网站，可以找到很多可以免费使用ip代理网站。但是免费的ip质量可能不太好，一是不能实现完全匿名，二是速度跟不上。当然，付费的代理ip质量会比较有保证。如果需要稳定的代理，最好购买。

## 如何使用代理

```
import requests
import json
import random
resp = requests.get("http://tor1024.com/static/proxy_pool.txt")
ips_txt = resp.text.strip().split("\n")
ips = []
for i in ips_txt:
    try:
        k = json.loads(i)
        ips.append(k)
    except Exception as e:
        print(e)
r = requests.get("http://bj.ganji.com/", proxies=random.choice(ips), timeout=6)
print(r)
```

这是使用代理的方法，调试的时候不用代理，等大量抓取的时候再去使用。

try...except 自己加一下。

## 代理ip不能使用，导致爬虫不能运行

很多同学在写爬虫时，常常会出现以下问题：requests.exceptions.ProxyError: .... Failed to establish a new connection: [WinError 10061] 如下：

```
requests.get('http://www.baidu.com', proxies=proxies)
File "D:\python3.4\lib\site-packages\requests\api.py", line 71, in get
    return request('get', url, params=params, **kwargs)
File "D:\python3.4\lib\site-packages\requests\api.py", line 57, in request
    return session.request(method=method, url=url, **kwargs)
File "D:\python3.4\lib\site-packages\requests\sessions.py", line 475, in request
    resp = self.send(prep, **send_kwargs)
File "D:\python3.4\lib\site-packages\requests\sessions.py", line 585, in send
    r = adapter.send(request, **kwargs)
File "D:\python3.4\lib\site-packages\requests\adapters.py", line 465, in send
    raise ProxyError("request=%request)
requests.exceptions.ProxyError: HTTPConnectionPool(host='60.191.157.156', port=3128): Max retries exceeded with url: http://www.baidu.com/ (Caused by ProxyError)
```

这时候不是程序哪里有问题，也不是网站不能访问，因为你使用的代理ip是不能使用的，那么这时你该怎么做呢？

```
requests.get("http://www.baidu.com", proxies=proxies)
```

- 1、将proxies=proxies去掉。。
- 2、换一个代理即可。

## 爬虫遇js加载解决办法一

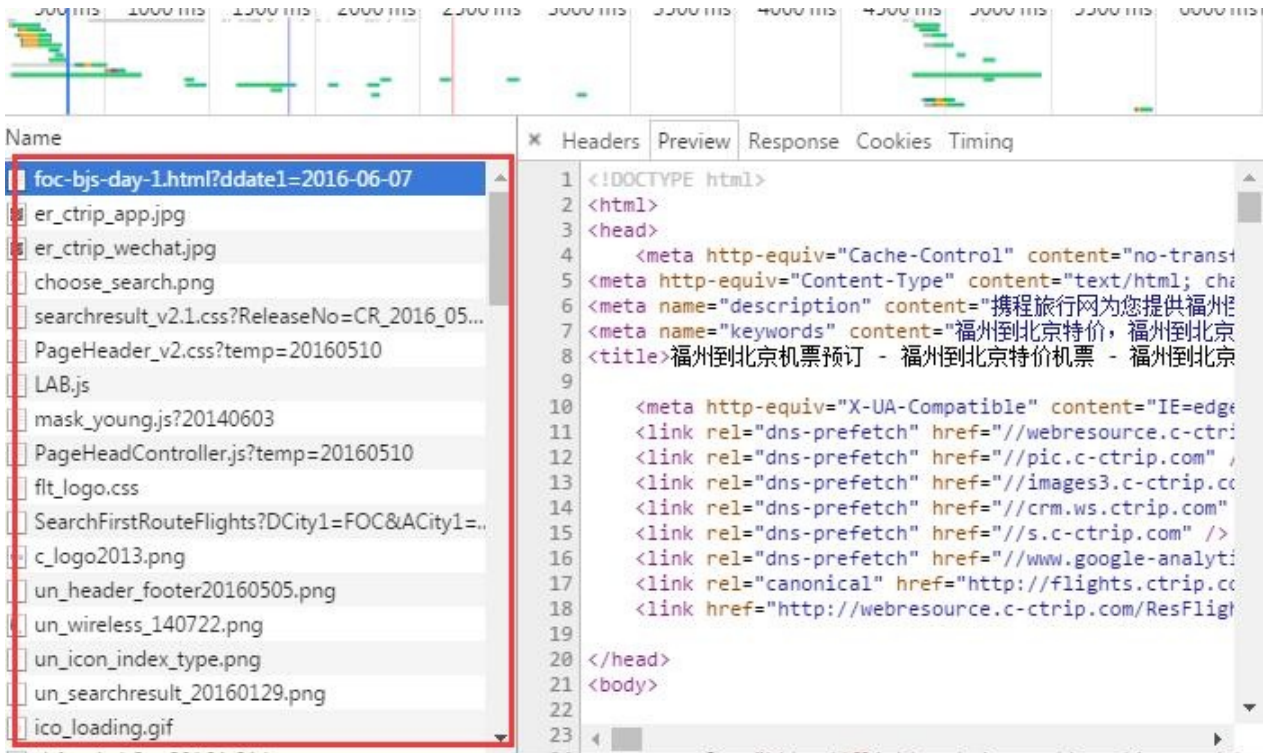
在使用爬虫爬取网页信息时，常常会遇到js脚本加载数据，近而导致无法正确地爬取数据的问题。在我们的教程中，已经有示范了一个例子，解决办法是使用浏览器伪装成浏览器。那么现在示范另一种爬虫常用的技巧。上图：



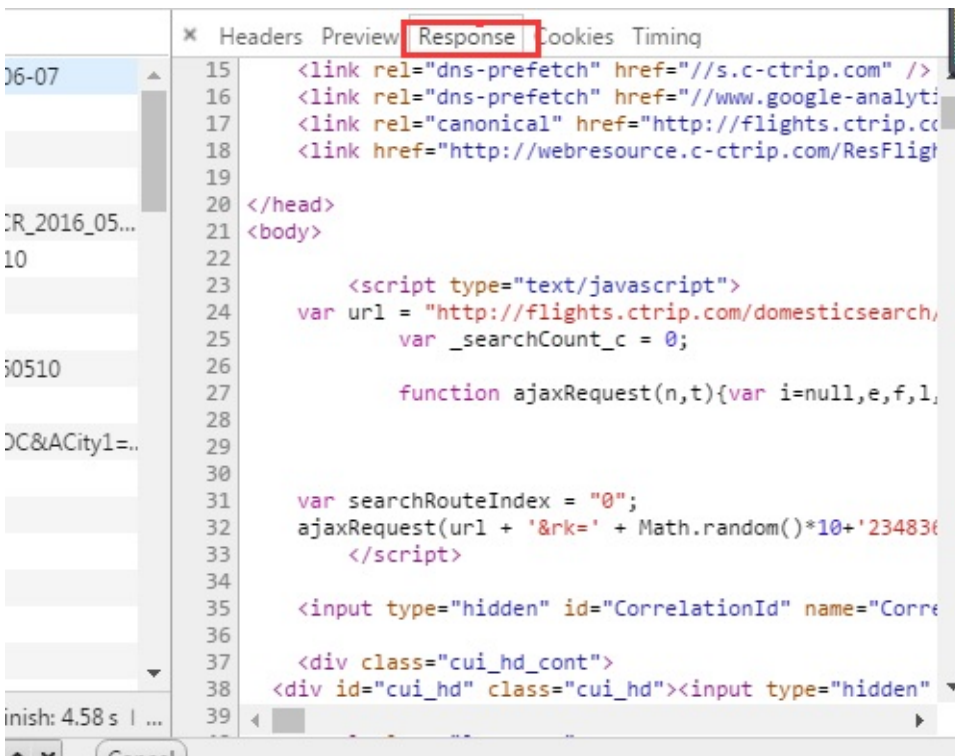
如图，我们如何使用爬虫来进行机票航班的查询呢？首先应该是看看机票查询时是将数据post到哪个网址，如图：

```
isSingleSearchPost" type="hidden">
<input id="SEOAirlineDibitCode" name=
"SEOAirlineDibitCode" type="hidden">
<div class="search_form" style="zoom:1;">...</div>
<div class="btn_box">
<input type="button" id="search_btn" data-ubt=
"Search_Flt" value="搜索" class="search" cdm=
"btn_submitsearch"> == $o
<input type="button" id="search_flthotel_btn" data-
ubt="Search_FltHtl" value="搜索机+酒" class=
"search_flthotel" cdm="btn_submitFlthotelSearch"
onclick="RedirectToFlthotelAction();">
</div>
```

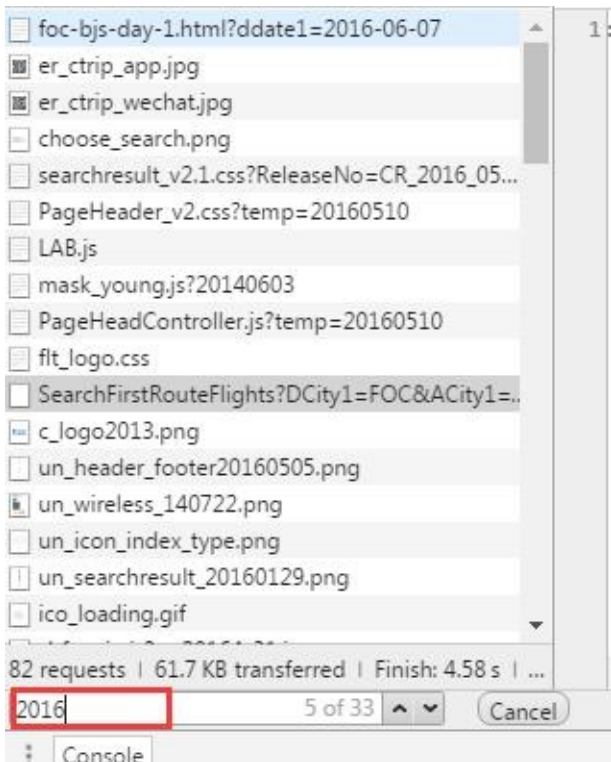
没找到我们想要的网址，这应该是使用js，当我们点击搜索按钮时，js脚本会将数据post上去，近而达到搜索的目的。那现在我们怎么才能知道post的网址呢？使用开发者工具，按下搜索，来看看网络是如何进行的：



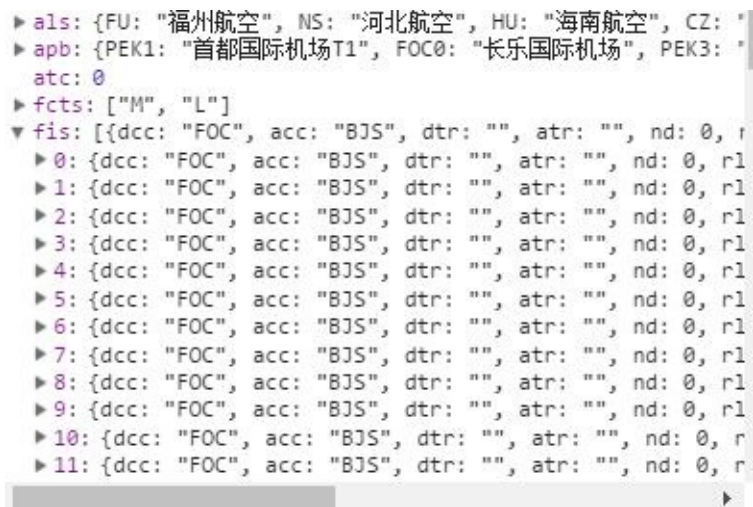
这么多，到底该看哪个呢？这时我也不知道选谁，那么多，那就看看谁比较适合自己呗。



最简单的方法就是把所有的response或者preview都看一遍，这时一般都可以看到你想要的数。但是总有不一般的时候，找不到怎么办，想想我们查询的数据有什么，这个数据常常都会出现在url中。那我们也搜索试试。



重点看看这几个有2016的数据，ok，找到了



那现在该回到我们最初的原点了 代码如下：

```
import requests
resp =
requests.get("http://flights.ctrip.com/domesticsearch/search/SearchFirstRouteFlights?
DCity1=FOC&ACity1=BJS&SearchType=S&DDate1=2016-06-07")
```

其中DCity1=FOC 代表出发城市，ACity1=BJS 代表目的城市，DDate1=2016-06-07 代表搜索日期。等等

至于该如何处理resp的内容，就根据返回来的内容的格式了，在这不多做解释，有兴趣的可以去研究研究。



# MongoDB 问题集



# 怎么导入.csv和.json文件数据

## 1. 导入json格式数据

### 问题描述

不会导入 json 格式到数据库啊，怎么办？

### 问题解答

#### --- Windows系统 ---

以管理员权限用 `cd` 命令进入MongoDB安装目录中 `bin` 目录下，运行如下命令：

```
D:\MongoDB>bin> mongoimport -d 数据库名字 -c 集合名字 --type 文件类型 --file 文件路径 --upsert
```

其中，`--upsert` 字段的意思是以插入(insert)或者更新(update)的方式来导入数据。

例子：

把路径为 `E:\mongodate\1202.json` 的json数据文件导入到MongoDB的 `test` 数据库中的 `MyJson` 集合中，那么命令可以这么写：

```
D:\MongoDB>bin> mongoimport -d test -c MyJson --type json --file E:\mongodate\1202.json --upsert
```

#### --- Mac OS X系统 ---

Mac系统区别就在于，不用 `cd` 命令进入MongoDB安装目录中 `bin` 目录下再运行，而是直接在终端运行上面所讲的命令即可。

## 2. 导入csv格式数据

---

### 问题衍生



# 安装MongoDB出现的错误集合

## 一、Permission denied

```
create/open lock file: /data/db/mongod.lock errno:13
Permission denied Is a mongod instance already running?, terminating
```

原因是Permission denied（权限拒绝），看来是当前用户执行mongod这个命令时，对/data/db这个目录没有操作权限，OK，知道原因就好办了，给/data/db加上权限。

在终端输入如下命令：

`sudo chown -R 当前登录的用户名 /data`

## 二、windows服务无法开启

1. 管理员权限启动命令行
2. `sc delete MongoDB`
3. 写好mongod.cfg，配置好数据库文件夹为C:/data/db，日志文件夹C:/data/log,这个文件放到C:/data/mongod.cfg
4. `cd /d "C:/Program Files/MongoDB/Server/3.2/bin"`
5. `mongod.exe --config "C:/data/mongod.cfg" --install 6.net start MongoDB`

## 三、

```
Requested option conflicts with current storage engine option for directoryPerDB;
you requested true but the current server storage is already set to false and cannot b
e changed, terminating
```

执行`mongod.exe --config "C:/data/mongod.cfg" --directoryperdb --install`时 去掉--directoryperdb

# 爬虫报错问题集

## 断点详解

当我们针对一个大的网站写好爬虫程序时，往往需要使用很多时间去爬取我们所需要的信息，因为一旦停止运行，所有保存在内存中的已经爬取过的链接以及其他数据将不存在，所以我们得保证我们的程序不会中途终止运行，否则将会前功尽弃。可是在爬取的过程中，我们很难能避开所有的错误使程序完美的运行。

---

在课程中，我们介绍了一种断点的方法，代码如下：

```
db_urls = [item['url'] for item in url_list.find()] #取出所有目标地址链接
index_urls = [item['url'] for item in item_info.find()]#取出已经爬取过的地址链接
x = set(db_urls)
y = set(index_urls)
rest_of_urls = x-y#还需爬取的地址链接
```

那么这段程序应该如何使用呢？将该程序放在抓取网页信息的程序段之前，并且只需执行一次即可，如下：

```
if name == 'main':
    db_urls = [item['url'] for item in ganji_url.find()]
    index_urls = [item['url'] for item in ganji_data1.find()]
    x = set(db_urls)
    y = set(index_urls)
    rest_of_urls = x - y

    pool = Pool()
    pool.map(get_item_info_from, rest_of_urls)
```

注意事项：1、请勿将断点程序直接加入我们爬取信息的函数中，这将使我们的程序多做无用功。2、在使用断点程序之前，请先将目标url，和已经爬取过的url存入数据库。

---

那么除了加上断点程序，还可以做些什么，来使我们的程序更有健壮性吗？答案是有的，那就是使用**try...except...**这是一个捕抓错误的程序，当问题出现时，捕抓它，并且处理，而不至于使我们的程序直接停止运行。代码如下：

```
def get_item_info_from(url, data=None):  
    try:  
        wb_data = requests.get(url)  
    except Exception as e:  
        print(e)
```

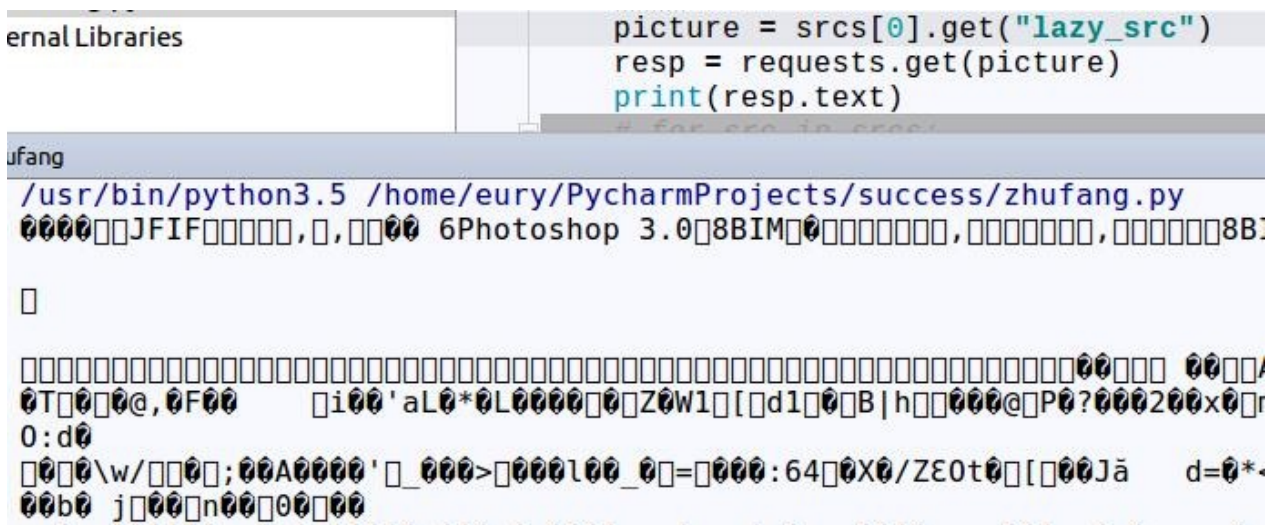
这样就可以避免我们在访问网站时，出现错误而导致程序停止运行的尴尬局面了。而**try...except...**的范围还可以继续扩大。

# 图片下载乱码

## 问题描述

为什么我图片文件打开看上去都是乱码呢？

```
resp = requests.get(image)
print(resp.text)
```



图片和我们平时看到的文字并不一样，文字有各种编码，如“utf-8”，“gbk”等。

图片没有那么多编码，图片是一种二进制文件，所以当我们使用get去获取我们的图片链接时，得到的是一堆二进制数据。

所以我们也就不像查看文字一样去查看它。

## 问题解决

我们需要将得到的文件以二进制的方式存储下来：

```
#coding:utf-8
import requests

resp = requests.get("http://image.3001.net/images/20160608/14653769669317.jpg")
with open("test.jpg", "wb") as img:
    img.write(resp.content)
```

以这种方法就可以将图片保存下来。`open()`中的“wb”就意味着是以二进制的方式写入。



## 多进程问题集

# pool线程池的使用

## 新建单一进程

```
#导入multiprocessing模块
import multiprocessing
import time
#定义一个简单的函数
def func(msg):
    print(msg)
    time.sleep(1)
if __name__ == "__main__":
    #将函数作为第一个参数传入Process中，注意函数不能是类里面的函数，只能是独立定义的
    #把args参数作为第二个参数传入Process中，注意下面例子中的args=("hello ", )是打包成了元组类型，可迭代，但是里面只有一个元素
    p = multiprocessing.Process(target=func, args=("hello ", ))
    p.start()
    p.join()
    print("Done!")
```

## 使用进程池

```
import multiprocessing
import time
#定义一个简单的函数
def func(msg):
    print(msg)
    time.sleep(1)
if __name__ == "__main__":
    #processes=4是指最多并发的进程数
    pool = multiprocessing.Pool(processes=4)
    #将函数作为第一个参数传入Process中，注意函数不能是类里面的函数，只能是独立定义的
    #把args参数作为第二个参数传入Process中
    for item in range(3):
        msg = 'hello ' + str(item)
        pool.apply_async(func, (msg, ))
    pool.close()
    #pool.join()是用来等待进程池中的worker进程执行完毕，防止主进程在worker进程结束前结束。但必pool.join()必须用在pool.close()之后
    pool.join()
    print("Done!")
```

而课程中使用的 `pool.map()` 其实道理是一样的，不过第二个参数一定要是可迭代的，相当于参数一个个传进去对应着前面的函数。

如以上代码可以这么改：

```
import multiprocessing
import time
def func(msg):
    print(msg)
    time.sleep(1)
if __name__ == "__main__":
    pool = multiprocessing.Pool(processes=4)
    msg_list = []
    for item in range(3):
        msg_list.append( 'hello ' + str(item) )
    pool.map(func,msg_list)
    pool.close()
    pool.join()
    print("Done!")
```

## 第四周课程问题集

# Django 问题集

## 内容概括

Django是Python语言实现的开源Web框架，采用的是MVC软件设计模式，即模型（ Model ），视图（ View ）和控制器（ Controller ）。

Django的主要目的是简便、快速的开发数据库驱动的网站。它强调代码复用,多个组件可以很方便的以“插件”形式服务于整个框架，Django有许多功能强大的第三方插件，你甚至可以很方便的开发出自己的工具包。这使得Django具有很强的可扩展性。它还强调快速开发和DRY(Do Not Repeat Yourself)原则。

由于Django是一个比较复杂的框架，所以对于Python初学者来说，使用还是有困难的，跟着视频和问题集的解答可以学到更多东西。

---

## 参考资料

1. [Django 1.8.2 中文文档](#)
2. [Django 1.9.2 英文文档](#)

## 重设 Admin 密码

Django框架中，童鞋们不管是忘记了Admin密码想找回，还是想重新设置，都可以通过以下步骤来解决：

---

### 第一种解决方法

运行 `python manage.py changepassword UserName`，会提示输入新密码两次。

```
(django) ~/ python manage.py changepassword Jason
Changing password for user 'Jason'
Password:
Password (again):
```

---

### 第二种解决方法

先进入python解释器

```
python manage.py shell
```

如下代码可进行密码修改

```
from django.contrib.auth.models import User
user = User.objects.get(username='Your name')
#重新设置密码
user.set_password('new_password')
user.save()
```

## 关于django数据流问题

在django中，数据如何加载，以及正确的显示在网页上呢？

首先大概的流程是这样子的：



在步骤1中，我们使用mongodb的查询程序如：

```
class ArtiInfo(Document):#继承Document类    des = StringField()
    title = StringField()
    scores = StringField()
    tags = ListField(StringField())
    meta = {'collection': 'arti_info3'}
```

使用

```
arti_info = ArtiInfo.objects
```

就可以将我们在ArtiInfo要查询的信息返回回来。这是我们已经完成了步骤1。接着开始我们的步骤2，例如我们要使用分页功能，那么这时就可以

```
from django.core.paginator import Paginator
```

在调用的类中使用：

```
limit = 4
paginator = Paginator(arti_info, limit)
loaded = paginator.page(1) #第一页
context = {'ArtiInfo': loaded}
```

这时我们就完成了步骤2的任务，接着开始我们的步骤3。将我们处理过后的程序丢给我们的模板处理：

```
render(request, 'index.html', context)
```

最后在我们的模板中取出数据：

```
item.title
```

这时模板就渲染好了，这时我们就完成了步骤**3**的渲染工作，将渲染好的网页回应给我们的客户端吧。

数据流的大概流程就是这样，数据的进和出类似。



# Django数据库连接问题

在Django中数据库基本连接方法

1、在setting文件中：

```
from mongoengine import connect
connect('website',host='127.0.0.1',port=27017)
```

其中website代表你的数据库名称

2、在models中：

```
from mongoengine import *

connect('website',host='127.0.0.1',port=27017)

class ArtiInfo(Document):
    url = StringField()
    kind = StringField()

    meta = {'collection':'arti_info'}
```

其中，website同样代表你的数据库名，而arti\_info代表你的集合名。

注意，容易造成失误的是写错数据库名和集合名，同学们在无法查询到数据时，首先看看自己的数据库名和集合名是否写对。

# Semantic UI 问题集

## semantic UI 那些文件在哪里？

只要下载相应的css，js文件就行，然后像课程视频那样，分别放在css文件夹，js文件夹里面。

文件下载链接：<https://github.com/AJKipper/PythonProjects/tree/master/muggle-coding-courses/week4/course3/semantic-web-package>

## 课程相关的实用资源

- **Python** 标准库文档
    - [官方Python 2.7.11英文文档](#)
    - [官方Python 3.5.1英文文档](#)
    - [Python 中文官方文档 \( 包含很多，自己查看 \)](#)
- 

- **BeautifulSoup4** 文档
    - [英文文档](#)
    - [中文文档版本一](#)
    - [中文文档版本二](#)
- 

- **Requests** 文档
    - [英文文档](#)
    - [中文文档](#)
- 

- **Django** 框架文档
    - [Django1.9官方英文文档](#)
    - [Django1.8.2中文文档](#)
- 

- **Jupyter** 文档
    - [Jupyter 官方英文文档](#)
- 

- **PyMongo** 文档
    - [PyMongo 官方英文文档](#)
- 

- **MongoDB** 文档&教程
    - [MongoDB 官方英文文档](#)
    - [MongoDB 中文手册](#)
    - [《MongoDB 权威指南》电子书下载](#)
    - [MongoDB 菜鸟教程](#)
-

