# 传智Python爬虫

**Python** 抓取 ⽃⻥直播**App** 妹⼦图

【课程简介】 作为一个新世纪有思想、有文化、有道德、时刻准备着的屌丝

男⻘年，在现今社会中，没事⼉上上⽹、逛逛YY、看看⽃⻥、翻翻美⼥图片

那是必不可少的；在Python爬⾍的世界⾥，这都不是事⼉；Http抓包数据分

析，模拟⼿机请求；⽃⻥的妹⼦，快到碗⾥来。

爬虫分为定向的和非定向的，搜索引擎就非定向的爬虫。爬虫功能强大到能到整个互联网进行爬取的时候，就是搜索引擎了。

## 1认识爬虫

### 为什么要学习爬虫

#### 学习目的以及需求

需求来自于：抓取的某个网站或者某个应用的内容，提取有用的价值

#### 实现手段

模拟用户在浏览器或者App应用上的操作，实现自动化的程序

#### 爬虫应用场景（利用爬虫能做什么？）

* 大家最熟悉的应用场景

抢票神器（360抢票器）

投票神器（微信朋友圈投票）

* 企业应用场景

咨询报告：https://zhuanlan.zhihu.com/p/20898635?refer=8dian

1、各种热门公司招聘中的职位数及月薪分布

2、对某个App的下载量跟踪

3、 饮食地图

还可以把男的排除掉，只看女的：

4、 票房预测

#### 总结

* 咨询报告（咨询服务行业）
* 抢票神器
* 投票神器
* 预测（股市预测、票房预测）
* 国民情感分析
* 社交关系网络
* 政府部门舆情监控 等等 ...

## 2爬虫基本原理

爬虫是模拟用户在浏览器或者App应用上的操作，把操作的过程、实现自动化的程序

当我们在浏览器中输入一个url后回车，后台会发生什么？比如说你输入https://www.baidu.com

简单来说这段过程发生了以下四个步骤：

1. 查找域名对应的IP地址。

浏览器首先访问的是DNS(Domain Name System,域名系统), dns的主要工作就是把域名转换成相应的IP地址

1. 向IP对应的服务器发送请求。
2. 服务器响应请求，发回网页内容。
3. 浏览器显示网页内容。

网络爬虫要做的，简单来说，就是实现浏览器的功能。通过指定url，直接返回给用户所需要的数据， 而不需要一步步人工去操纵浏览器获取。

浏览器是如何发送和接收这个数据呢？

HTTP协议（HyperText Transfer Protocol，超文本传输协议）目的是为了提供一种发布和接收HTML(HyperText Markup Language)页面的方法。

HTTPS（全称：Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。

## 使用urllib爬取数据

### 1. urlopen( )

打开一个url的方法，返回一个文件对象，然后可以进行类似文件对象的操作

>>> # 导入模块

>>> import urllib

>>>

>>> # 打开指定的url，就好比操作本地文件一样

>>> f = urllib.urlopen('http://www.baidu.com')

>>>

>>> #读取html页面的第一行

>>> firstLine = f.readline()

>>> firstLine

'<!DOCTYPE html><!--STATUS OK--><html><head><meta http-equiv="content-type" content="text/html;charset=utf-8"><meta http-equiv="X-UA-Compatible" content="IE=Edge"><meta content="always" name="referrer"><meta name="theme-color" content="#2932e1"><link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" /><link rel="search" type="application/opensearchdescription+xml" href="/content-search.xml" title="\xe7\x99\xbe\xe5\xba\xa6\xe6\x90\x9c\xe7\xb4\xa2" /><link rel="icon" sizes="any" mask href="//www.baidu.com/img/baidu.svg"><link rel="dns-prefetch" href="//s1.bdstatic.com"/><link rel="dns-prefetch" href="//t1.baidu.com"/><link rel="dns-prefetch" href="//t2.baidu.com"/><link rel="dns-prefetch" href="//t3.baidu.com"/><link rel="dns-prefetch" href="//t10.baidu.com"/><link rel="dns-prefetch" href="//t11.baidu.com"/><link rel="dns-prefetch" href="//t12.baidu.com"/><link rel="dns-prefetch" href="//b1.bdstatic.com"/><title>\xe7\x99\xbe\xe5\xba\xa6\xe4\xb8\x80\xe4\xb8\x8b\xef\xbc\x8c\xe4\xbd\xa0\xe5\xb0\xb1\xe7\x9f\xa5\xe9\x81\x93</title>\n'

说明:

urlopen的返回值是一个对象，可以用type()来查看，或者直接打印f

1. read( ) , readline( ) ,readlines( ) , fileno( ) , close( ) ：这些方法的使用方式与文件对象完全一样
2. info( )：返回一个httplib.HTTPMessage对象，表示远程服务器返回的头信息
3. getcode( )：返回Http状态码。如果是http请求，200请求成功完成;404网址未找到，2开头的表示正常，5开头的表示服务器挂了，3开头的是重定向。
4. geturl( )：返回请求的url
5. 如果urlopen( )不能够成功打开一个url，那么会产生一个异常，此时可以通过try来捕获然后处理

### 2. urlretrieve( )

urlretrieve方法将url定位到的html文件下载到你本地的硬盘中。

如果不指定filename，则会存为临时文件。

urlretrieve()返回一个二元组(filename, mine\_hdrs)

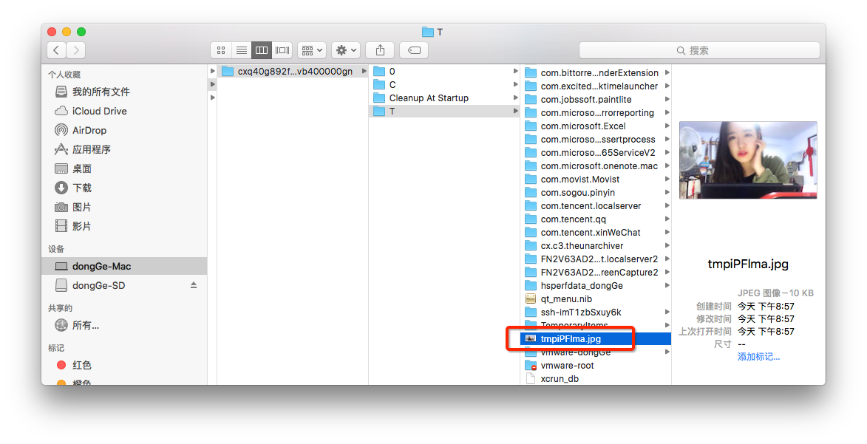
#### 2.1 不指定路径





ret = urllib.urlretrieve("https://www.----.jpg")

print(ret) #打印出来返回值



说明：

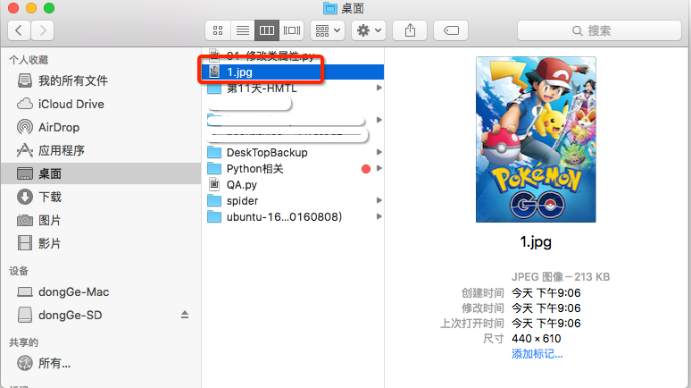
* 清除由于urllib.urlretrieve()所产生的缓存

#### 2.2 指定路径

ret = urllib.urlretrieve("https://www.----.jpg", "xiaoxiao.jpg")保存在当前位置下且重命名为xiaoxiao.jpg

ret = urllib.urlretrieve("https://www.----.jpg", filename="./xiaoxiao.jpg")





import urllib

response = urllib.urlopen("http://www.douyu.com")

response.getcode()

html = response.read()

startPoint = html.find("src")

EndPoint = html.find(".jpg")

html[startPoint:EndPoint+4]

#找到了第一个图片的地址

import re

images = re.findall(r"src='(.\*?\.jpg)' ",html) #是一个列表，这个方法把png和jpg混合到了一起。应该把png的也加上。

images = re.findall(r'src="(.\*?\.(jpg|png))" ',html) #返回一个列表，列表中的元素是元组，元组的第一个元素才是图片的地址

images[0]是第一个照片

urllib.urlretrieve(images[0],"/home/david/Desktop/0.jpg")

使用循环即可

import time

time.sleep(1)

#### 2.3 urlencode( )

上面的程序演示了最基本的网页抓取，不过，现在大多数网站都是动态网页，需要你动态地传递参数给它，它做出对应的响应。所以，在访问时，我们需要传递数据给它。最常见的情况是什么？对了，就是登录注册的时候呀。 把数据用户名和密码传送到一个URL，然后你得到服务器处理之后的响应，这个该怎么办？下面让我来为小伙伴们揭晓吧！ 数据传送分为POST和GET两种方式

两种方式有什么区别呢？

最重要的区别是GET方式是直接以链接形式访问，链接中包含了所有的参数，当然如果包含了密码的话是一种不安全的选择，不过你可以直观地看到自己提交了什么内容。

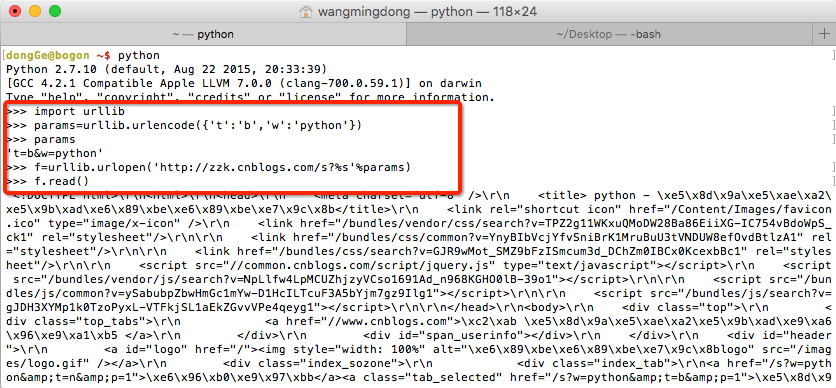
POST则不会在网址上显示所有的参数，不过如果你想直接查看提交了什么就不太方便了，大家可以酌情选择。







这里可以与urlopen结合以实现post方法和get方法：



url = <http://zzk.cnblogs.com/s?>

parament = urllib.urlencode({"t":"b","w":"ios"})

url + parament

如果路径很长，可以定义一个字典

kw={}

kw["t"] = "b"

kw["w"] = "ios"

urllib.urlencode(kw)

##### GET方法

>>> import urllib

>>> params=urllib.urlencode({'t':1,'eggs':2,'bacon':0})

>>> params

'eggs=2&bacon=0&spam=1'

>>> f=urllib.urlopen("http://python.org/query?%s" % params)

>>> print f.read()

有两种方式与服务器进行交互。输入网址回车时，默认是get的方式。urlopen是用get的方法。

##### POST方法

>>> import urllib

>>> parmas = urllib.urlencode({'spam':1,'eggs':2,'bacon':0})

>>> f=urllib.urlopen("http://python.org/query",parmas)

>>> f.read()

表单form，提交的数据过大时。网站的注册和登陆，都是post的方法。

url = <http://zzk,cnblogs.com>

urlopen(url)----->使用get的方法

xxxx = urlencode({"k": "b"})

urlopen(url,xxxx)----->post的方法

先用urlencode获得一个值，放在urlopen第二个参数中，就是使用post的方法。

## 使用urllib2爬取数据

刷新页面，在Network, Headers中，有General, Response Headers, Request Headers这三个信息往往就决定了能不能对这个网站进行爬取。

Response Headers返回的数据最前面的部分就是Response Headers。

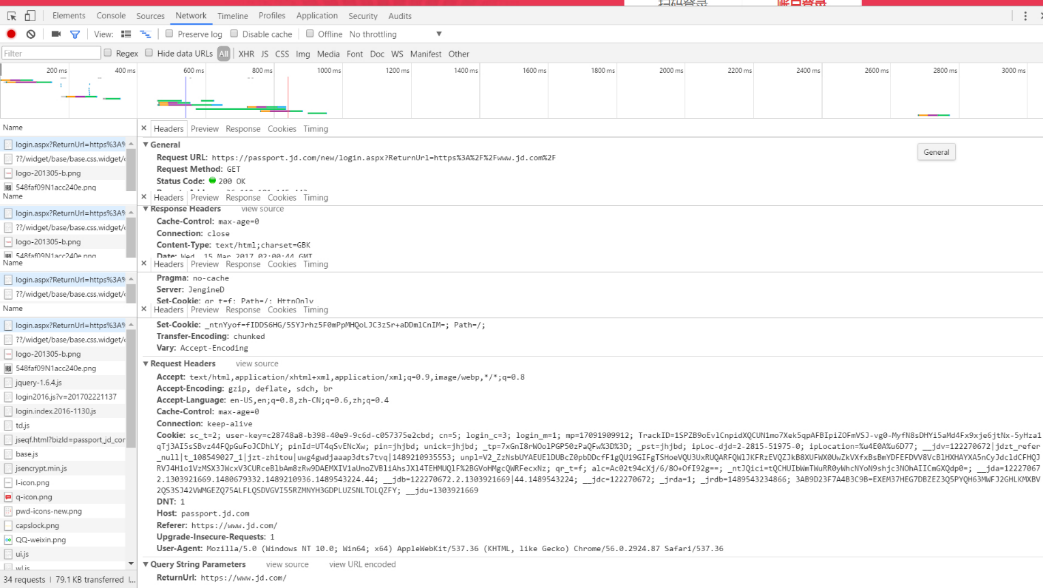
Request Headers是请求的时候发送的内容，其中就有用户的cookie。如果以后想要模拟一下登陆之后可以做的事情，如搜索的限制。

在登陆页面登陆一下，发送请求，获取网站Request Headers的内容，在使用程序爬取网站的内容时，把这部分本应该由浏览器提交的Request Headers的内容也进行提交，服务器就不能区分是爬虫的登陆还是浏览器用户的登陆了。

Set-Cookie: 用户是否登陆了网站。没有登陆时，发送的数据是缺少一部分数据，服务器检测到缺少这部分数据，就认为用户没有登陆。就返回登陆的页面。当用户登陆之后，生成一个随机的特别长的值，返回到你的浏览器中，这就是cookie的值。用户使用浏览器浏览网站时，浏览器都会把此cookie值进行提交，服务器就能够根据此值对用户进行区分。

在淘宝或京东搜索，浏览过一个商品时，再去访问其它有广告的网站，那浏览过的商品往往就会出现在广告中。也是由于cookie。即使是一个新的系统，新的浏览器，在浏览商品时，淘宝京东也会生成一个唯一的cookie值，并与浏览过的商品绑定。然后大数据分析，推送给阿里妈妈，给各大广告网站推送广告。现在在网站上，只要浏览了网站，就知道你做了什么。禁用cookie，防止浏览器和网站进行追踪。

总结：cookie是用来区分用户的。

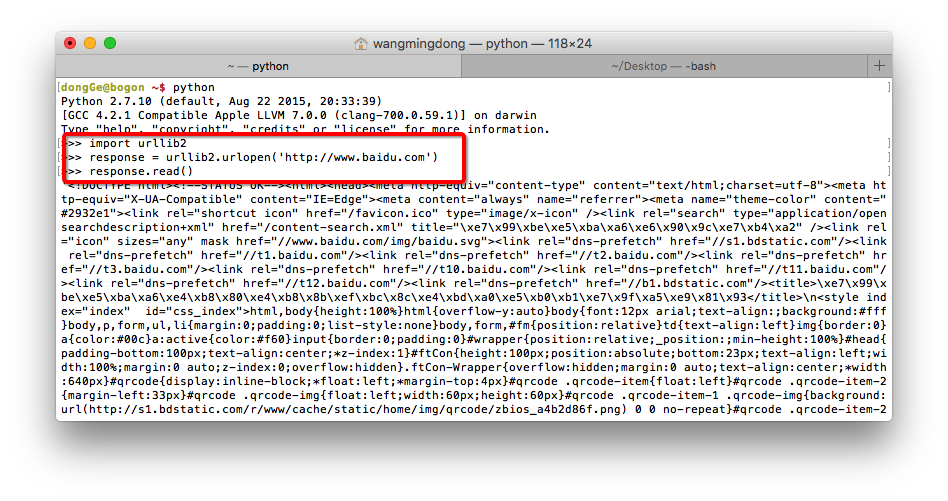


### 1. urlopen( )

import urllib2

response = urllib2.urlopen("http://www.baidu.com")

print response.read()



我们来分析这两行代码，第一行

response = urllib2.urlopen("http://www.baidu.com")

首先我们调用的是urllib2库里面的urlopen方法，传入一个URL，这个网址是百度首页，协议是HTTP协议，当然你也可以把HTTP换做FTP,FILE,HTTPS 等等，只是代表了一种访问控制协议

print response.read()

response对象有一个read方法，可以返回获取到的网页内容

如果不加read直接打印会是什么？答案如下：

<addinfourl at 139728495260376 whose fp = <socket.\_fileobject object at 0x7f1513fb3ad0>>

直接打印出了该对象的描述，所以记得一定要加read方法，否则它不出来内容可就不怪我咯！

### 2. 构造Requset

其实上面的urlopen参数可以传入一个request请求,它其实就是一个Request类的实例，构造时需要传入Url,Data等等的内容。比

如上面的两行代码，我们可以这么改写

import urllib2

request = urllib2.Request("http://www.baidu.com")

response = urllib2.urlopen(request)

print response.read()

运行结果是完全一样的，只不过中间多了一个request对象，推荐大家这么写，因为在构建请求时还需要加入好多内容，通过构建一个request，服务器响应请求得到应答，这样显得逻辑上清晰明确。

### 3. POST和GET

#### 3.1 GET方法

import urllib

import urllib2

values={}

values['username'] = "993484988@qq.com"

values['password']="XXXX"

data = urllib.urlencode(values)

url = "http://passport.csdn.net/account/login"

geturl = url + "?"+data

request = urllib2.Request(geturl)

response = urllib2.urlopen(request)

print response.read()

可以通过print geturl，打印输出一下url，发现其实就是原来的url加？然后加编码后的参数

http://passport.csdn.net/account/login?username=993484988%40qq.com&password=X

#### 3.2 POST方法

import urllib

import urllib2

values = {"username":"993484988@qq.com","password":"XXXX"}

data = urllib.urlencode(values)

url = "https://passport.csdn.net/account/login?from=http://my.csdn.net/my/mycsdn"

request = urllib2.Request(url,data)

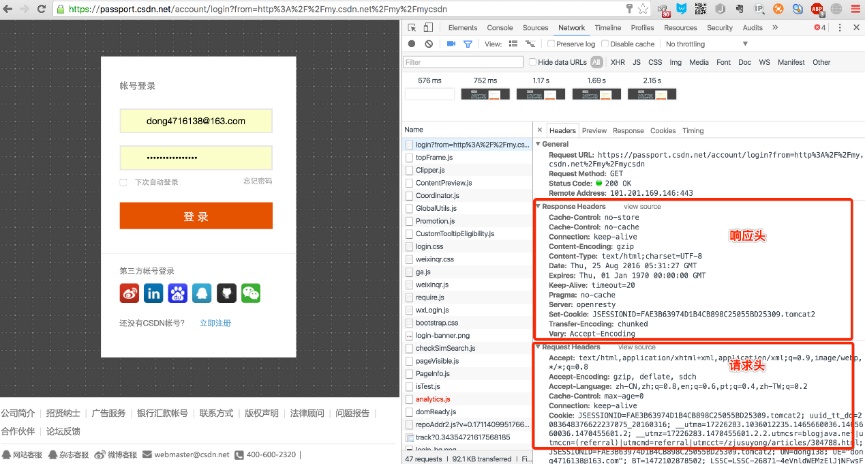
response = urllib2.urlopen(request)

print response.read()

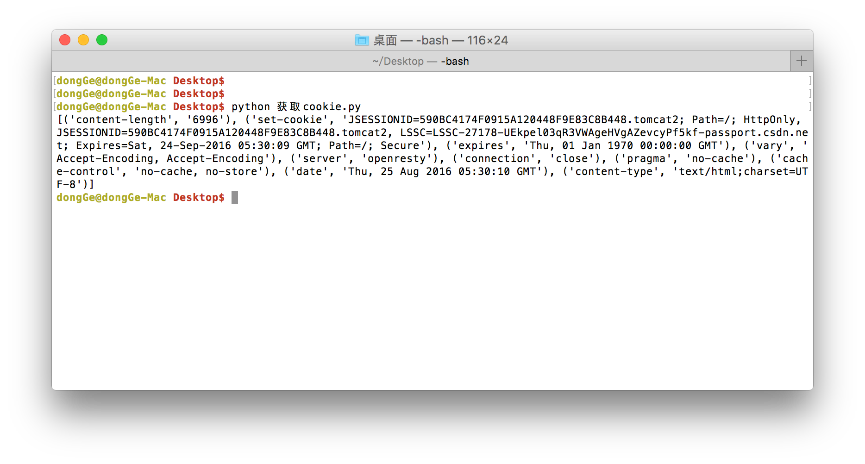
说明:

我们引入了urllib库，现在我们模拟登陆CSDN，当然上述代码可能登陆不进去，因为还要做一些设置头部header的工作，或者还有一些参数 没有设置全，还没有提及到在此就不写上去了，在此只是说明登录的原理。我们需要定义一个字典，名字为values，参数我设置了username和 password，下面利用urllib的urlencode方法将字典编码，命名为data，构建request时传入两个参数，url和data，运行程序，即可实现登陆，返回的便是登陆后呈现的页面内容。

##### 获取响应头信息







##### 伪造请求头信息

#coding=utf-8

import urllib2

import sys

#抓取网页内容-发送报头-1

url= "http://www.phpno.com"

send\_headers = {

'Host':'www.phpno.com',

'User-Agent':'Mozilla/5.0 (Windows NT 6.2; rv:16.0) Gecko/20100101 Firefox/16.0', 'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8','Connection':'keep-alive'

}

req = urllib2.Request(url,headers=send\_headers)

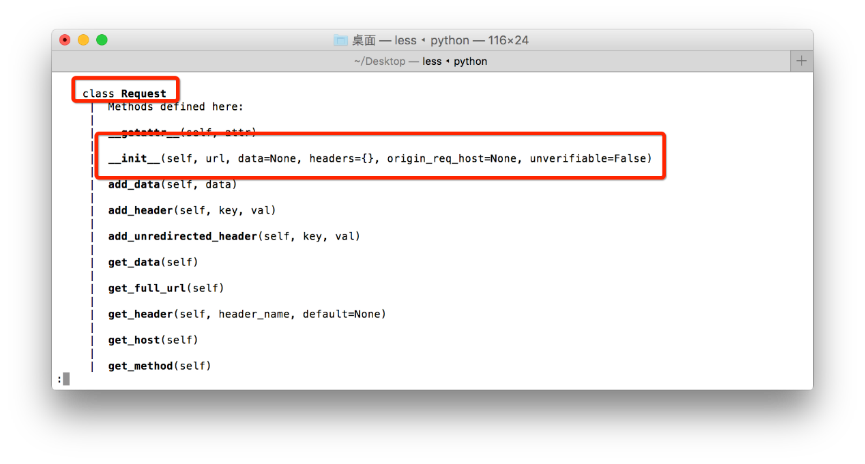
r = urllib2.urlopen(req)

...省略后面的代码...

通过 help(urllib2)可以查看这个模块中的很多类以及方法

除了延时外，还要伪装浏览器访问。服务器检测浏览器。

让自己的电脑实际访问一下，把提交的信息全部复制过来。服务器就不能区分了。



通过urlopen().info() 可以查看到很多的操作方法

### 4. urllib和urllib2的区别

* urllib2可以接受一个Request类的实例来设置URL请求的headers，urllib仅可以接受URL。这意味着，你不可以通过urllib模块伪装你的User Agent字符串等（伪装浏览器）。
* urllib提供urlencode方法用来GET查询字符串的产生，而urllib2没有。这是为何urllib常和urllib2一起使用的原因。
* urllib2模块比较优势的地方是urlliburllib2.urlopen可以接受Request对象作为参数，从而可以控制HTTP Request的header部。

## 使用简单的正则解析数据

#coding=utf-8

#正则表达式模块

import re

#获取路径模块

import urllib

def getHtml(url):

page=urllib.urlopen(url)

html=page.read()

return html

def getImag(html):

imglist = re.findall(r'src="(.\*?\.(jpg|png))"',html)

x=0

for imgurl in imglist:

print('正在下载 %s'imgurl[0])

urllib.urlretrieve(imgurl[0],'./downloads/%d.jpg'%x)

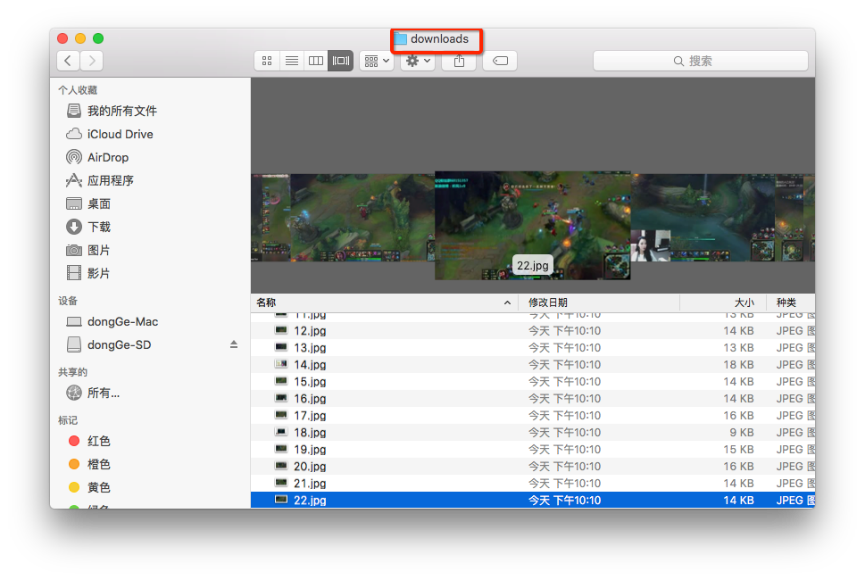
x+=1

加上个延时

time.sleep(1)

html=getHtml("http://www.douyu.com/directory/game/LOL")

getImag(html)



## 6使用 Beautiful Soup 解析数据

有的小伙伴们对写正则表达式的写法用得不熟练，没关系，我们还有一个更强大的工具，叫Beautiful Soup，有了它我们可以很方便地提取出HTML或XML标签中的内容，实在是方便，这一节就让我们一起来感受一下Beautiful Soup的魅力

### 1. Beautiful Soup的简介

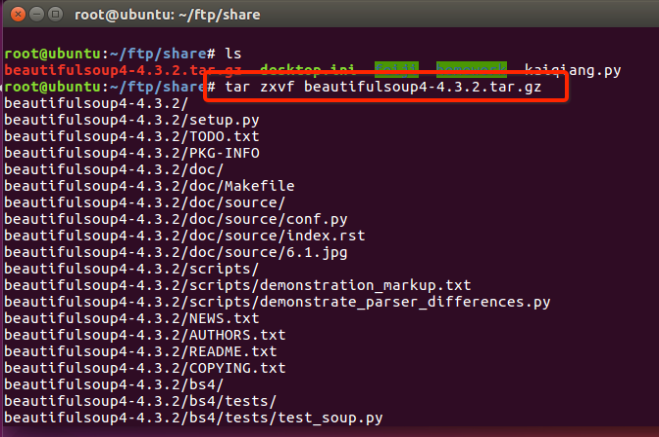
简单来说，Beautiful Soup是python的一个库，最主要的功能是从网页抓取数据。官方解释如下：

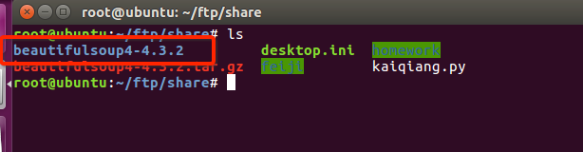
Beautiful Soup提供一些简单的、python式的函数用来处理导航、搜索、修改分析树等功能。它是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。 Beautiful Soup自动将输入文档转换为Unicode编码，输出文档转换为utf-8编码。你不需要考虑编码方式，除非文档没有指定一个编码方式，这时，Beautiful Soup就不能自动识别编码方式了。然后，你仅仅需要说明一下原始编码方式就可以了。 Beautiful Soup已成为和lxml、html6lib一样出色的python解释器，为用户灵活地提供不同的解析策略或强劲的速度。

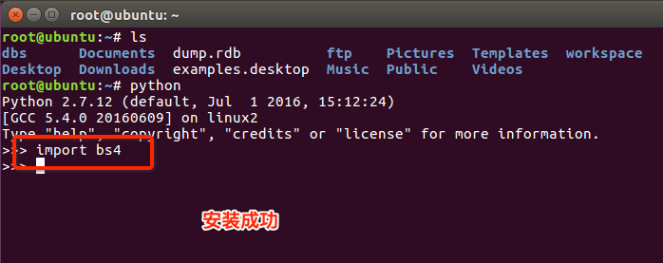
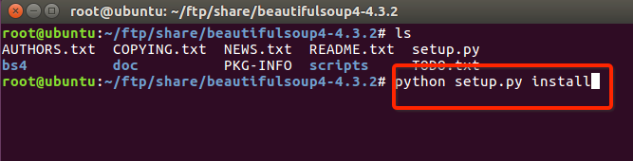
### 2. 安装

下载地址:https://pypi.python.org/pypi/beautifulsoup4/4.3.2

官方文档：http://beautifulsoup.readthedocs.org/zh\_CN/latest







### 3. 使用

from bs4 import BeautifulSoup

我们创建一个字符串，后面的例子我们便会用它来演示

html = """<html><head><title>The Dormouse's story</title></head>

<body>

<p class="title" name="dromouse"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were

<a href="http://example.com/elsie" class="sister" id="link1"><!-- Elsie --></a>,

<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and

<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;

and they lived at the bottom of a well.</p>

<p class="story">...</p>

"""

创建 beautifulsoup 对象

soup = BeautifulSoup(html)

下面我们来打印一下 soup 对象的内容，格式化输出

print(soup.prettify())

#### 3.1 找标签

直接打印标签

print soup.title

#<title>The Dormouse's story</title>

print soup.head

#<head><title>The Dormouse's story</title></head>

print soup.a

#<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>

print soup.p

#<p class="title" name="dromouse"><b>The Dormouse's story</b></p>

我们可以利用 soup加标签名轻松地获取这些标签的内容，是不是感觉比正则表达式方便多了？不过有一点是，它查找的是在所有内容中的第一个符合要求的标签

对于标签，它有两个重要的属性，是 name 和 attrs，下面我们分别来感受一下

print(soup.name)

print(soup.head.name)

#[document]

#head

soup 对象本身比较特殊，它的 name 即为 [document]，对于其他内部标签，输出的值便为标签本身的名称

print soup.p.attrs

#{'class': ['title'], 'name': 'dromouse'}

在这里，我们把 p 标签的所有属性打印输出了出来，得到的类型是一个字典。

如果我们想要单独获取某个属性，可以这样，例如我们获取它的 class 叫什么

print soup.p['class']

#['title']

#### 3.2 获取文字

既然我们已经得到了标签的内容，那么问题来了，我们要想获取标签内部的文字怎么办呢？很简单，用 .string 即可，例如

print soup.p.string

#The Dormouse's story

#### 3.3 CSS选择器

在CSS中，标签名不加任何修饰，类名前加点，id名前加 #，在这里我们也可以利用类似的方法来筛选元素，用到的方法是 soup.select()，返回类型是 list

##### 3.3.1 通过标签名查找

print soup.select('title')

#[<title>The Dormouse's story</title>]

##### 3.3.2 通过类名查找

print soup.select('.sister')

#[<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>]

##### 3.3.3 通过 id 名查找

print soup.select('#link1')

#[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>]

##### 3.3.4 组合查找

组合查找即和写 class 文件时，标签名与类名、id名进行的组合原理是一样的，例如查找 p 标签中，id 等于 link1的内容，二者需要用空格分开

print soup.select('p #link1')

#[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>]

##### 3.3.5 直接子标签查找

print soup.select("head > title")

#[<title>The Dormouse's story</title>]

##### 3.3.6 属性查找

查找时还可以加入属性元素，属性需要用中括号括起来，注意属性和标签属于同一节点，所以中间不能加空格，否则会无法匹配到。

print(soup.select('a[class="sister"]'))

#[<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>]

print soup.select('a[href="http://example.com/elsie"]')

#[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>]

同样，属性仍然可以与上述查找方式组合，不在同一节点的空格隔开，同一节点的不加空格

print soup.select('p a[href="http://example.com/elsie"]')

#[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>]

以上的 select 方法返回的结果都是列表形式，可以遍历形式输出，然后用 get\_text() 方法来获取它的内容。

soup = BeautifulSoup(html, 'lxml')

print type(soup.select('title'))

print soup.select('title')[0].get\_text()

for title in soup.select('title'):

print title.get\_text()

取标签中的值

tt = soup.a

tt["href"]

## 爬⾍项目说明书

### 评分规则

每组任意挑选题目，但至少有2个不一样的题目，每人都进⾏检查

不写注释 -3分

没有函数或者类 -5分

没有完成功能 -5分

用面向对象完成 +6分

用函数完成 +2

难度分数：难度等级\*5

其他加分项：只要是创新功能，每一项 +5分

其他减分项：没有代码、不备份等，每一项 -5分

每组的分数：所有人的平均值

分数不设上、下限

最终解释权归dongGe、huiGe所有

### 项目1：采集python职位（难度等级：13）

要求**:**

1. 采集出北京python职位的要求信息（信息数量不少于300条件）

2. 采集出的信息有：a. 薪资b. 公司名称c. 公司地址d. 职位描述

3. 把采集到的信息存放到1个txt中，格式不定，但要能够分清楚采集到的不

同的职位

### 项目2：采集苹果价格（难度等级：15）

要求**:**

1. 采集出苹果的价格信息（信息数量至少500条）

2. 采集出的信息有：a. 发布日期 b. 地区 c. 品种 d. 价格

3. 把采集到的信息存放到1个txt中，格式不定，但要能够分清楚采集到的不

同的职位

### 项目3：采集⾖瓣top250图书（难度等级：10）

要求**:**

1. 采集出top250图书信息（至少200条）

2. 采集出的信息有：a. 图书名称 b. 作者 c. 评分 d. 评分人数 e.价格

3. 把采集到的信息存放到1个txt中，格式不定，但要能够分清楚采集到的不

同的职位

### 项目4：采集搞笑图片（难度等级：11）

要求**:**

1. 采集出搞笑图片100张

2. 用图片的描述信息当做图片的名字

3. 把采集到搞笑图片存放到images文件夹中

form bs4 import Beautifulsoup

import urllib

response = urllib.urlopen("http://www.3jy.com/") #是一个对象，不允许直接进行操作。

html = response.read()

print(html) #只要其中有内容即读到了。或者进行200的判断。

type(html) #是一个字符串，不能进行操作

soup = BeautifulSoup(html) #创建beautiful对象

temp = sou.select("div") #是一个列表

#会获得很多的内容，必须再加上限定条件

len(temp)

joke1 = soup.select("div[id = content-1]") #是一个列表，不能直接用select对列表进行操作，必须要把列表中的元素提取出来。

len(joke1) #只有一个元素

jokeContent = joke1[0]

type(jokeContent) #是"bs4.element.Tag"类型的，即是bs4对象类型的，而不是普通的字符串，可以继续调用select进行选择。

jokeTitle = jokeContent("h2")

len(jokeTitle)

titleContent = jokeTitle[0]

type(titleContent) #还是个对象，还可以进行选择。

titleHtml = titleContent.select("a")

titleHtmlContent = titleHtml[0]

title = titleHtmlContent.string

print(title) #即获得了标题

获得图片的链接地址

jokeTitle = jokeContent("div[class=listpic]")

jokePic = jokeTitle[0]

jokePicImg = jokePic.select("img") #获取img标签

jokePicLink= jokePicImg[0]

PicLink = jokePicLink["src"]