Contents

[网络请求和响应 --requests 1](#_Toc458430347)

[7个http动作 --发送请求, 返回Response 对象 1](#_Toc458430348)

[为URL传递参数 1](#_Toc458430349)

[定制请求头 1](#_Toc458430350)

[响应对象 2](#_Toc458430351)

[保存响应内容 2](#_Toc458430352)

[解析HTML and XML --Beautiful Soup 3](#_Toc458430353)

[创建BeautifulSoup对象 3](#_Toc458430354)

[标签Tag对象 3](#_Toc458430355)

[Tag查询 4](#_Toc458430356)

[Tag美观打印 5](#_Toc458430357)

[常用场景： 5](#_Toc458430358)

网络爬虫

<Python编程快速上手 –让繁琐工作自动化>

Web抓取是指利用程序下载并处理来自web的内容

# 网络请求和响应 --requests

$pip install requests

Requests模块是“人类“用的http模块，类似的模块有urllib，urllib2，httplib，httplib2

## 7个http动作 --发送请求, 返回Response 对象

r = requests.get('https://github.com/timeline.json')

r = requests.post("http://httpbin.org/post")

r = requests.put("http://httpbin.org/put")

r = requests.delete("http://httpbin.org/delete")

r = requests.head("http://httpbin.org/get")

r = requests.options("http://httpbin.org/get")

## 为URL传递参数

payload = {'key1': 'value1', 'key2': 'value2'}

r = requests.get("http://httpbin.org/get", params=payload)

r.url == http://httpbin.org/get?key2=value2&key1=value1

## 定制请求头

url = 'https://api.github.com/some/endpoint'

payload = {'some': 'data'}

headers = {'content-type': 'application/json'}

r = requests.post(url, data=json.dumps(payload), headers=headers)

表单形式的POST请求

payload = {'key1': 'value1', 'key2': 'value2'}

r = requests.post("http://httpbin.org/post", data=payload)

POST一个多部分编码(Multipart-Encoded)的文件

url = 'http://httpbin.org/post'

files = {'file': open('report.xls', 'rb')}

r = requests.post(url, files=files)

Basic Authentication

requests.get('https://api.github.com/user', auth=('user', 'pass'))

OAuth 1 Authentication

auth = OAuth1('YOUR\_APP\_KEY', 'YOUR\_APP\_SECRET',

'USER\_OAUTH\_TOKEN', 'USER\_OAUTH\_TOKEN\_SECRET')

requests.get(url, auth=auth)

Proxies

proxies = {

'http': 'http://165.225.96.34:10015',

'https': ' http://165.225.96.34:10015',

}

requests.get('http://example.org', proxies=proxies)

## 响应对象

r.headers 响应头

r.status\_code == requests.codes.ok 响应状态码

r.cookies Cookies

r.text 响应内容

r.content 二进制响应内容

from PIL import Image

from StringIO import StringIO

i = Image.open(StringIO(r.content))

r.json JSON响应内容

r.raw 原始响应内容

## 保存响应内容

with open(filename, 'wb') as fd:

for chunk in r.iter\_content(chunk\_size):

fd.write(chunk)

# 解析HTML and XML --Beautiful Soup

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

$pip install beautifulsoup4

$pip install lxml HTML解析器 速度快

$pip install html5lib 解析方式与浏览器相同

from bs4 import BeautifulSoup

## 创建BeautifulSoup对象

soup = BeautifulSoup(open("index.html"))

soup = BeautifulSoup("<html>data</html>")

Beautiful Soup将复杂HTML文档转换成一个复杂的树形结构,每个节点都是Python对象,所有对象可以归纳为4种: Tag , NavigableString , BeautifulSoup , Comment .

指定文档解析器

如果仅是想要解析HTML文档,只要用文档创建 BeautifulSoup 对象就可以了，Beautiful Soup会自动选择一个解析器来解析文档.但是还可以通过参数指定使用那种解析器来解析当前文档.

BeautifulSoup 第一个参数应该是要被解析的文档字符串或是文件句柄,第二个参数用来标识怎样解析文档.如果第二个参数为空,那么Beautiful Soup根据当前系统安装的库自动选择解析器,解析器的优先数序: lxml, html5lib, Python标准库.在下面两种条件下解析器优先顺序会变化:

要解析的文档是什么类型: 目前支持, “html”, “xml”, 和 “html5”

指定使用哪种解析器: 目前支持, “lxml”, “html5lib”, 和 “html.parser”

编码

任何HTML或XML文档都有自己的编码方式,比如ASCII 或 UTF-8,但是使用Beautiful Soup解析后,文档都被转换成了Unicode:

Beautiful Soup用了编码自动检测子库来识别当前文档编码并转换成Unicode编码

编码自动检测功能大部分时候都能猜对编码格式,但有时候也会出错.有时候即使猜测正确,也是在逐个字节的遍历整个文档后才猜对的,这样很慢.如果预先知道文档编码,可以设置编码参数来减少自动检查编码出错的概率并且提高文档解析速度.在创建 BeautifulSoup 对象的时候设置 from\_encoding 参数.

## 标签Tag对象

Tag 对象与XML或HTML原生文档中的tag相同:

soup = BeautifulSoup('<b class="boldest">Extremely bold</b>')

tag = soup.b

tag中最重要的属性: name和attributes

str(tag) tag -> str

tag.name == 'b'

tag.attrs 字典

tag['class']

tag.string == u'Extremely bold'

unicode\_string = unicode(tag.string)

tag的 .contents tag的子节点 （列表）

tag.contents[0]

tag.descendants tag的所有子孙节点 （列表）

tag.strings 子孙字符串 （列表）

tag.stripped\_strings 去除多余空白的子孙字符串 （列表）

tag.parent 元素的父节点

tag.parents 元素的所有父辈节点 (列表）

tag.next\_sibling 和 .previous\_sibling 属性来查询兄弟节点

tag.next\_siblings 和 .previous\_siblings 属性可以对当前节点的兄弟节点迭代输出

tag.get\_text() 方法,这个方法获取到tag中包含的所有文版内容包括子孙tag中的内容

## Tag查询

find\_all( name , attrs , recursive , text , \*\*kwargs )

soup.a 第一个<a>标签:

soup.find\_all('a') 所有的<a>标签

soup.find\_all(["a", "b"]) 所有<a>标签和<b>标签

soup.find\_all(id='link2')

soup.find\_all("a", attrs={"class": "sister"})

soup.find\_all(text=re.compile("Dormouse"))

soup.select("p nth-of-type(3)") 基于css选择器方法，返回Tag列表

Tag创建与替换

创建一个tag最好的方法是调用工厂方法 BeautifulSoup.new\_tag() :

tag.append(new\_tag)

PageElement.replace\_with() 方法移除文档树中的某段内容,并用新tag或文本节点替代它

PageElement.wrap() 方法可以对指定的tag元素进行包装

## Tag美观打印

prettify() 方法将Beautiful Soup的文档树格式化后以Unicode编码输出,每个XML/HTML标签都独占一行

## 常用场景：

从文档中找到所有<a>标签的链接

for link in soup.find\_all('a'):

print(link.get('href'))

# http://example.com/elsie

# http://example.com/lacie

# http://example.com/tillie

从文档中获取所有文字内容

print(soup.get\_text())

# The Dormouse's story

#

# ...

下载所有XKCD漫画

import requests, os, bs4

url ='http://xkcd.com'

os.makedirs('xkcd', exist\_ok=True)

while not url.endswith('#'):

res = requests.get(url)

res.raise\_for\_status() #如果下载文件出错，抛出异常

soup = bs4.BeautifulSoup(res.text)

comicElem = soup.select('#comic img')

if comicElem == []:

print('Could not find comic image.')

else:

comicUrl = 'http:' + comicElem[0].get('src')

res = requests.get(comicUrl)

res.raise\_for\_status()

即使页面是纯文本的，你也需要写入二进制数据，而不是文本数据，目的是为了保存该文本中的"unicode编码"

imageFile = open(os.path.join('xkcd', os.path.basename(comicUrl)), 'wb')

for chunk in res.iter\_content(100\*1000):

imageFile.write(chunk)

imageFile.close()

#Get the Prev button's url

prevLink = soup.select('a[rel="prev"]')[0]

url = 'http://xkcd.com' + prevLink.get('href')

用requests的响应text创建BeautifulSoup会有中英文乱码问题，建议使用如下

import urllib2, bs4

html = urllib2.urlopen(url).read()

soup = bs4.BeautifulSoup(html, 'lxml')

import webbrowser

webbrowser.open('http://inventwithpython.com/') 启动浏览器，打开url

selenium模块让Python直接控制浏览器，实际点击链接，填写登录信息, 与Requests and Beautiful Soup相比，Selenium允许你用高级得多的方式与网页交互。但因为它启动了Web浏览器，假如你只是想从网络上下载一些文件，会有点慢，并且难以在后台运行

from selenium import webdriver

browser = webdriver.Firefox()

browser.get('http://inventwithpython.com')

查找

browser.find\_elements\_by\_css\_selector(selector)

如果页面上没有元素匹配该方法要查找的元素，selenium模块就会抛出NoSuchElement异常

点击

linkElem = browser.find\_element\_by\_link\_text('Read It Online')

linkElem.click()

填写并提交表单

browser.get('http://gmail.com')

emailElem = browser.find\_element\_by\_id('Email')

passwordElem = browser.find\_element\_by\_id('Passwd')

emailElem.send\_keys('lin.qizhong@gmail.com')

passwordElem.send\_keys('...')

passwordElem.submit() 在任何元素上调用submit()方法，等同于该元素所在表单的submit按钮

from selenium.webdriver.common.keys import Keys

htmlElem = browser.find\_element\_by\_tag\_name('html')

htmlElem.send\_keys(Keys.END)

htmlElem.send\_keys(Keys.HOME) 按下home键，从而页面滚动至顶