

# 爬虫代理小记与aiohttp代理尝试



treelake (/u/66f24f2c0f36) + 关注

2017.04.06 22:43\* 字数 1805 阅读 1562 评论 7 喜欢 51 赞赏 1

(/u/66f24f2c0f36)

总结了一些爬虫代理的资料和知识，并尝试使用asyncio和aiohttp使用代理ip访问目标网站，按代理IP的访问效果实时更新代理IP得分，初始获取3000左右代理IP，在稳定后，对摩拜单车信息的访问可以达到40次/秒-100次/秒。

## 代理IP方案简述

- 快速抓取网站时，要应对IP每分钟访问次数有限制甚至会封IP的服务器，使用代理IP可以帮助我们。
- Kaito (<http://kaito-kidd.com/>)的爬虫代理服务 (<http://kaito-kidd.com/2015/11/02/proxies-service/>)讲得很清楚，推荐。
- Github上现成的开源代理爬虫也不少，如：
  - qiyeboy (<https://github.com/qiyeboy>)/**IPProxyPool** (<https://github.com/qiyeboy/IPProxyPool>)
  - jhao104 (<https://github.com/jhao104>)/**proxy\_pool** ([https://github.com/jhao104/proxy\\_pool](https://github.com/jhao104/proxy_pool))
  - awolfly9 (<https://github.com/awolfly9>)/**IPProxyTool** (<https://github.com/awolfly9/IPProxyTool>)
  - fancoo (<https://github.com/fancoo>)/**Proxy** (<https://github.com/fancoo/Proxy>)
  - derekhe (<https://github.com/derekhe>)/mobike-crawler (<https://github.com/derekhe/mobike-crawler>)/modules (<https://github.com/derekhe/mobike-crawler/tree/master/modules>)/**ProxyProvider.py** (<https://github.com/derekhe/mobike-crawler/blob/master/modules/ProxyProvider.py>)
- 思路也都很清楚，从更多的代理网站上爬取免费代理，存入自己的数据库，定时更新。在拿到代理IP后，验证该代理的有效性。并且提供简单的API来获取代理IP。
- 七夜的博客python开源IP代理池--IPProxys (<http://www.cnblogs.com/qiyeboy/p/5693128.html>)详细地阐释了自己的代码。
- 大部分的代理网站的爬取还是比较简单的，在上述开源的代码中包含了不少代理网站的爬取与解析。困难点的有js反爬机制，也都被用selenium操作无头webkit或者js代码解析以及python的js代码执行库所解决。此外有趣的是，在上面的开源代码中出现了用爬取得到的代理来访问代理网站的情况。
- 定时刷新代理，有自定义的代理定时刷新模块，也可用celery定时任务。
- 验证有效性的方式有：
  - 直接访问百度。
  - 访问 <http://icanhazip.com/> (<http://icanhazip.com/>)，得到返回的IP。
  - 访问 [http://httpbin.org/get?show\\_env=1](http://httpbin.org/get?show_env=1) ([http://httpbin.org/get?show\\_env=1](http://httpbin.org/get?show_env=1))，得到访问头的详细信息，判断代理的匿名程度。



- 访问其他目标网站，如豆瓣等。
- API的提供可以用BaseHTTPServer拓展下，也可用简便的flask或者Django加上插件提供restful api服务。
- 对于免费的代理IP来说，最重要的一是量大，就算有很大比例无效的，还是能拿到一些高质量的代理。一个网站的未筛选代理能有几千个，多个网站就很可观了，当然要考虑到重复。
- 再就是代理IP的筛选机制，不少开源库都添加了评分机制，这是非常重要的。例如利用对累计超时次数以及成功率的加权来评判代理IP的质量(<https://github.com/fancoo/Proxy#%E7%AD%96%E7%95%A5>)。在每次使用后都对代理IP的情况进行评价，以此来刷新数据库，方便下一次选取优质的代理IP。
- 如何对代理IP进行评价，在成功和失败的各种情况中如何奖惩，筛选出最优质的代理IP是非常重要的。
- 此外，每个代理IP的使用也要考虑是否要设置一定的使用间隔，避免过于频繁导致失效。

## 尝试

- 自然，首先要做的就是从免费代理网站上获取大量代理IP，我选择了最方便的66ip，接口很简单，一次性访问可以拿到3000左右的代理，当然，频繁访问会导致js反爬机制，这时再简单地使用selenium+phantomJs即可。

```
url = ("http://m.66ip.cn/mo.php?tqsl={proxy_number}")
url = url.format(proxy_number=10000)
html = requests.get(url, headers=headers).content
html = html.decode(charset.detect(html)['encoding'])
pattern = r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}:\d{1,5}'
all_ip = re.findall(pattern, html)
```

- 然后就是设置代理IP的奖惩机制，我参考了摩拜单车爬虫源码及解析(<http://www.jianshu.com/p/8662de6f0d7a>)使用类，看起来简单清晰，每个代理IP对象拥有自己的ip与分数，@property 令它们可以被点操作符访问。初始分数为100分，这里分数都弄成整数。如果代理成功，按照延时的大小来给予奖励，即调用代理的相应方法，最高10分，而超时扣10分，连接错误扣30分，其它错误扣50分（可以酌情修改）。为了便于代理IP之间的比较，修改了\_\_lt\_\_方法。



```
class Proxy:
    def __init__(self, ip):
        self._url = 'http://' + ip
        self._score = 100

    @property
    def url(self):
        return self._url

    @property
    def score(self):
        return self._score

    def __lt__(self, other):
        '''
        由于优先队列是返回最小的，而这里分数高的代理优秀
        所以比较时反过来
        '''
        return self._score > other._score

    def success(self, time):
        self._score += int(10 / int(time + 1))

    def timeoutError(self):
        self._score -= 10

    def connectError(self):
        self._score -= 30

    def otherError(self):
        self._score -= 50
```

- 感觉上，最好的验证方式是直接访问目标网站。除去根本不能用的一部分，代理IP的有效性对不同的目标网站是有区别的，在我的尝试中，豆瓣相比摩拜对代理IP的应对明显更好，这里代码为访问摩拜。在第一轮的筛选中，对每个代理IP，访问两次目标网站，超时时间为10秒，依情况奖惩，保留分数大于50的。总共花费22秒左右时间，排除了一小部分根本不能使用的代理，也对所有代理的分数初步更新。



```

async def douban(proxy, session):
# 使用代理访问目标网站，并按情况奖惩代理
    try:
        start = time.time()
        async with session.post(mobike_url,
                                data=data,
                                proxy=proxy.url,
                                headers=headers, # 可以引用到外部的headers
                                timeout=10) as resp:

            end = time.time()
            # print(resp.status)
            if resp.status == 200:
                proxy.success(end - start)
                print('%6.3d' % proxy._score, 'Used time-->', end - start, 's')
            else:
                proxy.otherError()
                print('*****', resp.status, '*****')
    except TimeoutError as te:
        print('%6.3d' % proxy._score, 'timeoutError')
        proxy.timeoutError()
    except ClientConnectionError as ce:
        print('%6.3d' % proxy._score, 'connectError')
        proxy.connectError()
    except Exception as e:
        print('%6.3d' % proxy._score, 'otherError-->', e)
        proxy.otherError()
# ClientHttpProxyError

# TCPConnector维持链接池，限制并行连接的总量，当池满了，有请求退出再加入新请求，500和100相差不大
# ClientSession调用TCPConnector构造连接，Session可以共用
# Semaphore限制同时请求构造连接的数量，Semaphore充足时，总时间与timeout差不多

async def initDouban():

    conn = aiohttp.TCPConnector(verify_ssl=False,
                                limit=100, # 连接池在windows下不能太大
                                use_dns_cache=True)

    tasks = []
    async with aiohttp.ClientSession(loop=loop, connector=conn) as session:
        for p in proxies:
            task = asyncio.ensure_future(douban(p, session))
            tasks.append(task)

        responses = asyncio.gather(*tasks)
        await responses
    conn.close()

def firstFilter():
    for i in range(2):
        s = time.time()
        future = asyncio.ensure_future(initDouban())
        loop.run_until_complete(future)
        e = time.time()
        print('----- init time %s-----\n' % i, e - s, 's')

    num = 0
    pq = PriorityQueue()
    for proxy in proxies:
        if proxy._score > 50:
            pq.put_nowait(proxy)
            num += 1
    print('原始ip数:%s' % len(all_ip), '; 筛选后:%s' % num)
    return pq

```

- 然后就是正式的访问了，这里我使用了基于堆的asyncio优先队列（非线程安全）。通过asyncio.Semaphore限制并发请求连接的数量，不断地从队列中拿取最优质的代理IP，在访问结束后再将它放回队列。结果是，多个连接不会同时使用一个代理IP，如果代理成功，它将会很快被放回队列，再次使用。（如果需要设置成功代理的使用间隔，可以改为在访问成功后，先释放连接与信号量，然后使用asyncio.sleep(x)等待一段时间再放入优先队列，如果在genDouban函数里实现，可设置为range(concurrency)一定程度上大于 Semaphore(concurrency)）奖惩一直进行，一开始会有一段筛选的过程，稳定后的输出如下：



```
473 Used time--> 0.42412686347961426 s
171 Used time--> 0.16487789154052734 s
-014 Used time--> 0.67523193359375 s
-049 Used time--> 8.096510648727417 s
035 Used time--> 3.064735174179077 s
123 Used time--> 0.5208511352539062 s
143 Used time--> 0.5619871616363525 s
093 Used time--> 0.11780023574829102 s
-045 Used time--> 2.642759323120117 s
191 Used time--> 0.400421142578125 s
483 Used time--> 0.12423300743103027 s
-029 Used time--> 0.9408984184265137 s
103 Used time--> 0.11173439025878906 s
-014 Used time--> 7.914304971694946 s
089 Used time--> 1.683044195175171 s
-050 connectError
150 Used time--> 0.3608975410461426 s
493 Used time--> 0.16594171524047852 s
-018 Used time--> 1.213590383529663 s
113 Used time--> 0.11079573631286621 s
023 Used time--> 0.44400691986083984 s
163 Used time--> 0.6750969886779785 s
-020 Used time--> 0.7894964218139648 s
310 Used time--> 0.516232967376709 s
020 Used time--> 3.698575019836426 s
160 Used time--> 0.10678339004516602 s
123 Used time--> 0.10982155799865723 s
062 Used time--> 0.48073410987854004 s
065 Used time--> 0.5052714347839355 s
027 Used time--> 0.5098135471343994 s
069 Used time--> 0.5659432411193848 s
170 Used time--> 0.10528063774108887 s
-049 Used time--> 8.513077974319458 s
503 Used time--> 0.22108840942382812 s
-039 Used time--> 0.49025893211364746 s
-049 Used time--> 8.538584232330322 s
071 Used time--> 0.8480677604675293 s
133 Used time--> 0.5228219032287598 s
-032 Used time--> 0.9287652969360352 s
-027 Used time--> 5.821086406707764 s
016 Used time--> 0.9016931056976318 s
-035 Used time--> 0.5824878215789795 s
-004 Used time--> 0.6723277568817139 s
-027 Used time--> 7.544523477554321 s
-007 Used time--> 0.8422365188598633 s
045 Used time--> 0.7154984474182129 s
-050 connectError
180 Used time--> 0.31550025939941406 s
015 Used time--> 1.274921178817749 s
206 Used time--> 2.352935314178467 s
320 Used time--> 0.508458137512207 s
074 Used time--> 1.8834965229034424 s
080 Used time--> 1.9597609043121338 s
-037 timeoutError
-039 timeoutError
-050 timeoutError
-050 timeoutError
-049 otherError-->
072 Used time--> 0.49747467041015625 s
181 Used time--> 0.9591214656829834 s
173 Used time--> 0.6809825897216797 s
079 Used time--> 0.5852339267730713 s
190 Used time--> 0.28981637954711914 s
-015 Used time--> 5.936133861541748 s
143 Used time--> 0.5220968723297119 s
-008 Used time--> 0.8108019828796387 s
-019 Used time--> 0.9827635288238525 s
-049 Used time--> 9.164980173110962 s
200 Used time--> 0.10229611396789551 s
453 Used time--> 3.1340854167938232 s
```



```

pq = firstFilter()

async def genDouban(sem, session):
    # Getter function with semaphore.
    while True:
        async with sem:
            proxy = await pq.get()
            await douban(proxy, session)
            await pq.put(proxy)

async def dynamicRunDouban(concurrency):
    ...,
    TCPConnector维持链接池，限制并行连接的总量，当池满了，有请求退出再加入新请求
    ClientSession调用TCPConnector构造连接，Session可以共用
    Semaphore限制同时请求构造连接的数量，Semaphore充足时，总时间与timeout差不多
    ...,

    conn = aiohttp.TCPConnector(verify_ssl=False,
                                limit=concurrency,
                                use_dns_cache=True)

    tasks = []
    sem = asyncio.Semaphore(concurrency)

    async with aiohttp.ClientSession(loop=loop, connector=conn) as session:
        try:
            for i in range(concurrency):
                task = asyncio.ensure_future(genDouban(sem, session))
                tasks.append(task)

            responses = asyncio.gather(*tasks)
            await responses
        except KeyboardInterrupt:
            print('-----finishing-----\n')
            for task in tasks:
                task.cancel()
            if not conn.closed:
                conn.close()

    future = asyncio.ensure_future(dynamicRunDouban(200))
    loop.run_until_complete(future)

```

- 最后，我们中断程序，查看下代理IP的得分情况：

```

scores = [p.score for p in proxies]
scores.sort(reverse=True)
print('Most popular IPs:\n ----- \n', scores[:50],
      [i for i in scores if i > 100])
loop.is_closed()

```

```

Most popular IPs:
-----
[1221, 1040, 874, 831, 819, 724, 596, 578, 571, 501, 474, 474, 469, 452, 397,
387, 326, 323, 319, 293, 286, 284, 282, 281, 256, 249, 249, 243, 240, 239, 238,
236, 226, 225, 221, 215, 214, 207, 205, 201, 200, 195, 194, 187, 176, 175, 173,
157, 153, 153] [1221, 1040, 874, 831, 819, 724, 596, 578, 571, 501, 474, 474,
469, 452, 397, 387, 326, 323, 319, 293, 286, 284, 282, 281, 256, 249, 249, 243,
240, 239, 238, 236, 226, 225, 221, 215, 214, 207, 205, 201, 200, 195, 194, 187,
176, 175, 173, 157, 153, 153, 151, 151, 150, 150, 149, 146, 141, 139, 135, 135,
135, 129, 128, 126, 126, 123, 123, 123, 123, 120, 115, 114, 113, 112, 112, 111, 110,
109, 107, 102]

```

### 其它方案概览：

- 在Scrapy官方文档 (<https://doc.scrapy.org/en/latest/topics/practices.html?highlight=proxy#avoiding-getting-banned>)避免被封的建议提到了Tor - 洋葱路由 (<https://www.wikiwand.com/zh-hans/Tor>)：

```

use a pool of rotating IPs. For example, the free Tor project
(https://www.torproject.org/) or paid services like ProxyMesh
(http://proxymesh.com/). An open source alterantive is scrapoxy
(http://scrapoxy.io/), a super proxy that you can attach your own proxies to.

```

- 在知乎python 爬虫 ip池怎么做？ (<https://www.zhihu.com/question/47464143>)的回答中，提到了Squid ([https://www.wikiwand.com/zh-hans/Squid\\_cache](https://www.wikiwand.com/zh-hans/Squid_cache))与修改x-



forward-for标签的方法：

1. 使用squid的cache\_peer机制，把这些代理按照一定格式(具体格式参考文档)写入到配置文件中，配置好squid的端口，那么squid就可以帮你调度代理了，而且还可以摒弃失效的代理。
2. 在访问的http request里添加x-forward-for标签client随机生成，宣称自己是一台透明代理服务器

- 在如何突破豆瓣爬虫限制频率？(<https://www.v2ex.com/t/260777>)中提到：

用带 bid（可以伪造）的 cookie 去访问 - github  
(<https://github.com/dontcontactme/doubanspiders/blob/master/douban/album/misc/middlewares.py>)

## 其他资料

- Making 1 million requests with python-aiohttp  
(<https://pawelhmh.github.io/asyncio/python/aiohttp/2016/04/22/asyncio-aiohttp.html>)
- AsyncIO for the Working Python Developer (<https://hackernoon.com/asyncio-for-the-working-python-developer-5c468e6e2e8e>)
- How to crawl a quarter billion webpages in 40 hours  
(<http://www.michaelnielsen.org/ddi/how-to-crawl-a-quarter-billion-webpages-in-40-hours/>)

## 代码



```

from selenium import webdriver
import time
import aiohttp
from aiohttp.client_exceptions import ClientConnectionError
from aiohttp.client_exceptions import TimeoutError
import asyncio
from asyncio.queue import PriorityQueue
import chardet
import re
import requests
from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

headers = {'User-Agent': ('Mozilla/5.0 (Windows NT 10.0; Win64; x64) '
                          'AppleWebKit/537.36 (KHTML, like Gecko) ')}
loop = asyncio.get_event_loop()

class Proxy:
    def __init__(self, ip):
        self._url = 'http://' + ip
        self._score = 100

    @property
    def url(self):
        return self._url

    @property
    def score(self):
        return self._score

    def __lt__(self, other):
        '''
        由于优先队列是返回最小的，而这里分数高的代理优秀
        所以比较时反过来
        '''
        return self._score > other._score

    def success(self, time):
        self._score += int(10 / int(time + 1))

    def timeoutError(self):
        self._score -= 10

    def connectError(self):
        self._score -= 30

    def otherError(self):
        self._score -= 50

def getProxies():
    url = ("http://m.66ip.cn/mo.php?tsql={proxy_number}")
    url = url.format(proxy_number=10000)
    html = requests.get(url, headers=headers).content
    html = html.decode(chardet.detect(html)['encoding'])
    pattern = r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}:\d{1,5}'
    all_ip = re.findall(pattern, html)
    if len(all_ip) == 0:
        driver = webdriver.PhantomJS(
            executable_path=r'D:/phantomjs/bin/phantomjs.exe')
        driver.get(url)
        time.sleep(12) # js等待5秒
        html = driver.page_source
        driver.quit()
        all_ip = re.findall(pattern, html)
    with open('66ip_' + str(time.time()), 'w', encoding='utf-8') as f:
        f.write(html)
    return all_ip

all_ip = set(getProxies()) | set(getProxies())
proxies = [Proxy(proxy) for proxy in all_ip]

mobike_url = "https://mwx.mobike.com/mobike-api/rent/nearbyBikesInfo.do"
data = { # 请求参数: 纬度, 经度!
        'latitude': '33.2',
        'longitude': '113.4',
    }
headers = {
    'referer': "https://servicewechat.com/",
}

async def douban(proxy, session):

```





```

try:
    start = time.time()
    async with session.post(mobike_url,
                           data=data,
                           proxy=proxy.url,
                           headers=headers, # 可以引用到外部的headers
                           timeout=10) as resp:

        end = time.time()
        # print(resp.status)
        if resp.status == 200:
            proxy.success(end - start)
            print('%6.3d' % proxy._score, 'Used time-->', end - start, 's')
        else:
            proxy.otherError()
            print('*****', resp.status, '*****')
except TimeoutError as te:
    print('%6.3d' % proxy._score, 'timeoutError')
    proxy.timeoutError()
except ClientConnectionError as ce:
    print('%6.3d' % proxy._score, 'connectError')
    proxy.connectError()
except Exception as e:
    print('%6.3d' % proxy._score, 'otherError-->', e)
    proxy.otherError()
# ClientHttpProxyError

# TCPConnector维持链接池, 限制并行连接的总量, 当池满了, 有请求退出再加入新请求, 500和100相差不大
# ClientSession调用TCPConnector构造连接, Session可以共用
# Semaphore限制同时请求构造连接的数量, Semaphore充足时, 总时间与timeout差不多

async def initDouban():

    conn = aiohttp.TCPConnector(verify_ssl=False,
                               limit=100, # 连接池在windows下不能太大, <500
                               use_dns_cache=True)

    tasks = []
    async with aiohttp.ClientSession(loop=loop, connector=conn) as session:
        for p in proxies:
            task = asyncio.ensure_future(douban(p, session))
            tasks.append(task)

        responses = asyncio.gather(*tasks)
        await responses
    conn.close()

def firstFilter():
    for i in range(2):
        s = time.time()
        future = asyncio.ensure_future(initDouban())
        loop.run_until_complete(future)
        e = time.time()
        print('----- init time %s-----\n' % i, e - s, 's')

    num = 0
    pq = PriorityQueue()
    for proxy in proxies:
        if proxy._score > 50:
            pq.put_nowait(proxy)
            num += 1
    print('原始ip数:%s' % len(all_ip), '; 筛选后:%s' % num)
    return pq

pq = firstFilter()

async def genDouban(sem, session):
    # Getter function with semaphore.
    while True:
        async with sem:
            proxy = await pq.get()
            await douban(proxy, session)
            await pq.put(proxy)

async def dynamicRunDouban(concurrency):
    ...
    TCPConnector维持链接池, 限制并行连接的总量, 当池满了, 有请求退出再加入新请求
    ClientSession调用TCPConnector构造连接, Session可以共用
    Semaphore限制同时请求构造连接的数量, Semaphore充足时, 总时间与timeout差不多
    ...

    conn = aiohttp.TCPConnector(verify_ssl=False,
                               limit=concurrency,
                               use_dns_cache=True)

    tasks = []

```



```

sem = asyncio.Semaphore(concurrency)

async with aiohttp.ClientSession(loop=loop, connector=conn) as session:
    try:
        for i in range(concurrency):
            task = asyncio.ensure_future(genDouban(sem, session))
            tasks.append(task)

        responses = asyncio.gather(*tasks)
        await responses
    except KeyboardInterrupt:
        print('-----finishing-----\n')
        for task in tasks:
            task.cancel()
        if not conn.closed:
            conn.close()

future = asyncio.ensure_future(dynamicRunDouban(200))
loop.run_until_complete(future)

scores = [p.score for p in proxies]
scores.sort(reverse=True)
print('Most popular IPs:\n ----- \n', scores[:50],
      [i for i in scores if i > 100])
loop.is_closed()

```

- 访问百度

```

async def baidu(proxy):
    ...
    验证是否可以访问百度
    ...
    async with aiohttp.ClientSession(loop=loop) as session:
        async with session.get("http://baidu.com",
                               proxy='http://' + proxy,
                               timeout=5) as resp:
            text = await resp.text()
            if 'baidu.com' not in text:
                print(proxy,
                      '\n---\nis bad for baidu.com\n')
                return False
            return True

```

- 访问icanhazip

```

async def testProxy(proxy):
    ...
    http://aiohttp.readthedocs.io/en/stable/client_reference.html#aiohttp.ClientSession.request
    ...
    async with aiohttp.ClientSession(loop=loop) as session:
        async with session.get("http://icanhazip.com",
                               proxy='http://' + proxy,
                               timeout=5) as resp:
            text = await resp.text()
            if len(text) > 20:
                return
            else:
                if await baidu(proxy):
                    firstFilteredProxies.append(proxy)
                # print('原始:', proxy, '; 结果:', text)

```

- 访问HttpBin



```
async def httpbin(proxy):
    """
    访问httpbin获取headers详情，注意访问https 代理仍为http
    参考资料: https://imququ.com/post/x-forwarded-for-header-in-http.html
    http://www.cnblogs.com/wentthink/p/HTTP_Proxy_TCP_Http-Headers_Check.html
    """
    async with aiohttp.ClientSession(loop=loop) as session:
        async with session.get("https://httpbin.org/get?show_env=1",
                               proxy='http://' + proxy,
                               timeout=4) as resp:
            json_ = await resp.json()
            origin_ip = json_['origin']
            proxy_ip = json_['headers']['X-Forwarded-For']
            via = json_['headers'].get('Via', None)
            print('原始IP:', origin_ip,
                  '; 代理IP:', proxy_ip,
                  '---Via:', via)
            if proxy_ip != my_ip and origin_ip == proxy_ip:
                annoy_proxies.append(proxy)
```

• 访问豆瓣API

```
async def douban(proxy):

    async with aiohttp.ClientSession(loop=loop) as session:
        try:
            async with session.get(('https://api.douban.com/v2/movie/top250'
                                   '?count=10'),
                                proxy='http://' + proxy,
                                headers=headers,
                                timeout=4) as resp:

                print(resp.status)
        except TimeoutError as te:
            print(proxy, te, 'timeoutError')
        except ClientProxyConnectionError as pce:
            print(proxy, pce, 'proxyError')
        except ClientConnectionError as ce:
            print(proxy, ce, 'connectError')
```

• 循环访问豆瓣导致暂时被封IP

```
headers = {'User-Agent': ('Mozilla/5.0 (Windows NT 10.0; Win64; x64) '
                           'AppleWebKit/537.36 (KHTML, like Gecko) ')}

while True:
    r = requests.get('http://douban.com', headers=headers)
    print(r.status_code)
    r = requests.get('https://movie.douban.com/j/search_subjects?'
                     'type=movie&tag=%E8%B1%86%E7%93%A3%E9%AB%9',
                     headers=headers)
    print(r.status_code)
```

Python (/nb/7231458)

举报文章    © 著作权归作者所有



**treelake** (/u/66f24f2c0f36)

+ 关注

写了 132432 字，被 2291 人关注，获得了 2352 个喜欢  
(/u/66f24f2c0f36)

无名之辈







♡ 喜欢 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-like-button) | 51

更多分享

(http://cwb.assets.jianshu.io/notes/images/1098576)

被以下专题收入，发现更多相似内容

爬虫专题 (/c/3e3636c40c41?utm\_source=desktop&utm\_medium=notes-

-  @IT·互联网 (/c/V2CqjW?utm\_source=desktop&utm\_medium=notes-included-collection)
-  程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)
-  生活不易 我用... (/c/8c01bfa7b98a?utm\_source=desktop&utm\_medium=notes-included-collection)
-  爬虫搜集 (/c/5d76c1faa12f?utm\_source=desktop&utm\_medium=notes-included-collection)
-  爬虫 (/c/53152ad70e88?utm\_source=desktop&utm\_medium=notes-included-collection)
-  Ip代理 (/c/dd5cde59d2c8?utm\_source=desktop&utm\_medium=notes-included-collection)
- 展开更多 ▾