Intro to Python web crawler

Xiao Zhou

Sunneversets Studio

2nd March 2019

Overview

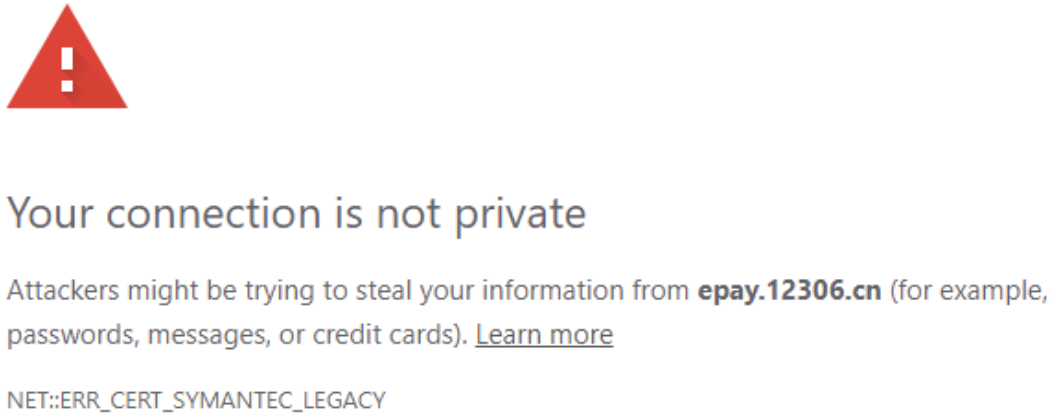
- HTTP basic
- Web basic
- Web crawler basic
- Requests
- Beautiful Soup
- Where to go?

Our keyword

1. our web pages are based HTTP, HTTPS
2. use URL to make request
3. use User-Agent to pretend to be like a browser
4. request return response
5. we get information from the response.

HTTP and HTTPS

- Protocol include http, https, ftp, sftp, smb …
- In web crawler, we usually crawl http and https
- HTTP : Hyper Text Transfer Protocol (超文本传输协议). transfer hypertext efficiently.
- HTTPS: (safer version of HTTP).
- However: when we want to pay for the ticket on https://www.12306.cn/ , we get error:

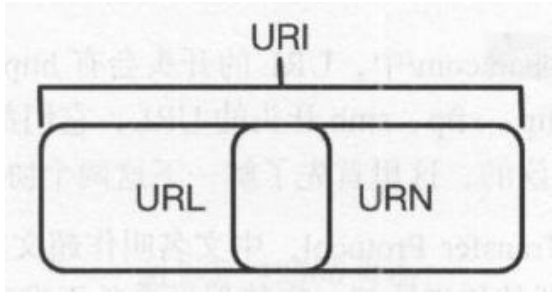


Our keyword

1. our web pages are based HTTP, HTTPS
2. use URL to make request
3. use User-Agent to pretend to be like a browser
4. request return response
5. we get information from the response.

URI & URL

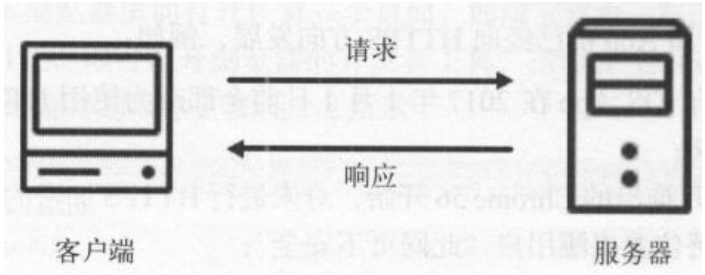
- URI: Uniform Resource Identifier (统一资源标志符)
- URL: Universal Resource Locator (统一资源定位符)
- They define the access method, including access protocol, path and name.
  e.g. https://github.com/favicon.ico (protocol: https, path: /, name: favicon.ico)
- Their relationship:



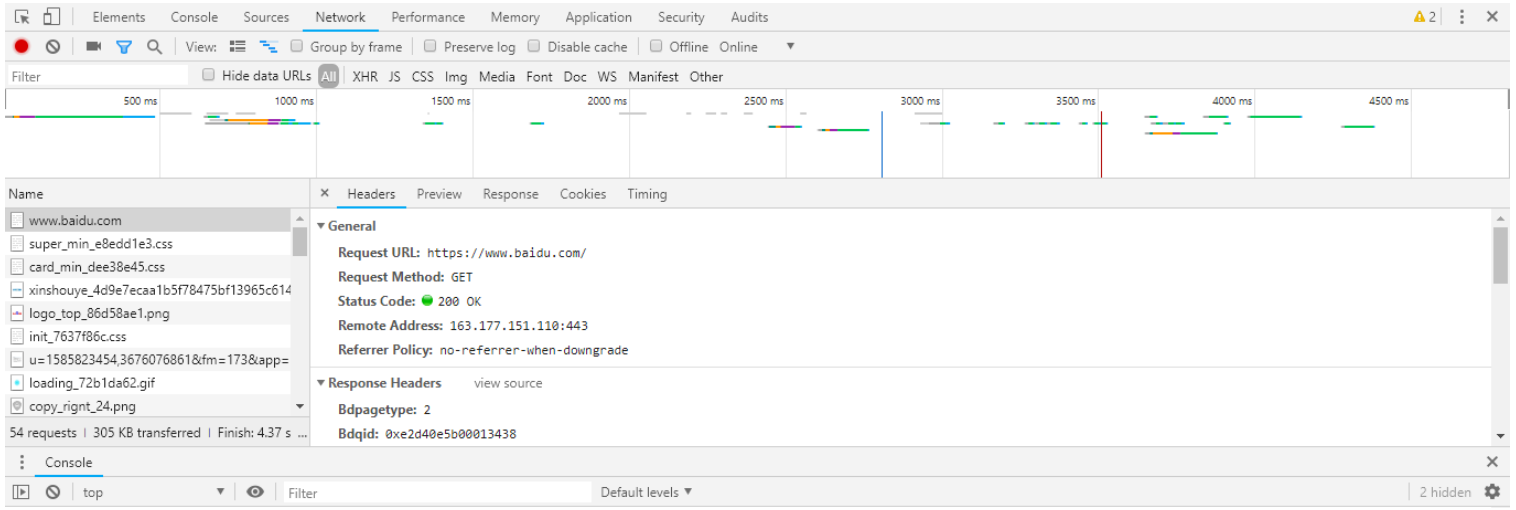
URN: Universal Resource Name, (统一资源名称)

HTTP request

1.model:



2. example: chrome browser -> F12 -> Network -> type https://www.baidu.com in chrome browser -> Name: www.baidu.com -> Headers



Our keyword

1. our web pages are based HTTP, HTTPS
2. use URL to make request
3. use User-Agent to pretend to be like a browser
4. request return response
5. we get information from the response.

Request

1. **Request Method:** usually get, post
2. **Request URL**
3. **Request header:** Cookies, User-Agent and …

Our keyword

1. our web pages are based HTTP, HTTPS
2. use URL to make request
3. use User-Agent to pretend to be like a browser
4. request return response
5. we get information from the response.

Response

1. response status code.

   e.g. 200 means ok, 404 means page not found, 500 means server error. and so on…

2. response headers

3. response body

   3.1 when you request a website, the body will be HTML.

   3.2 when you request a picture, the body will be binary.

Our keyword

1. our web pages are based HTTP, HTTPS
2. use URL to make request
3. use User-Agent to pretend to be like a browser
4. request return response
5. we get information from the response.

Web crawler basic

- What is a web crawler?
- Its workflow?

requests **Beautiful Soup**

get web pages -> fetch information -> save data.

requests

- How to install it?

  pip3 install requests

- Test code:

```
import requests

r = requests.get("https://www.baidu.com")
print(type(r))
print(r.status_code)
print(type(r.text))
print(r.text)
```

requests.get(url)

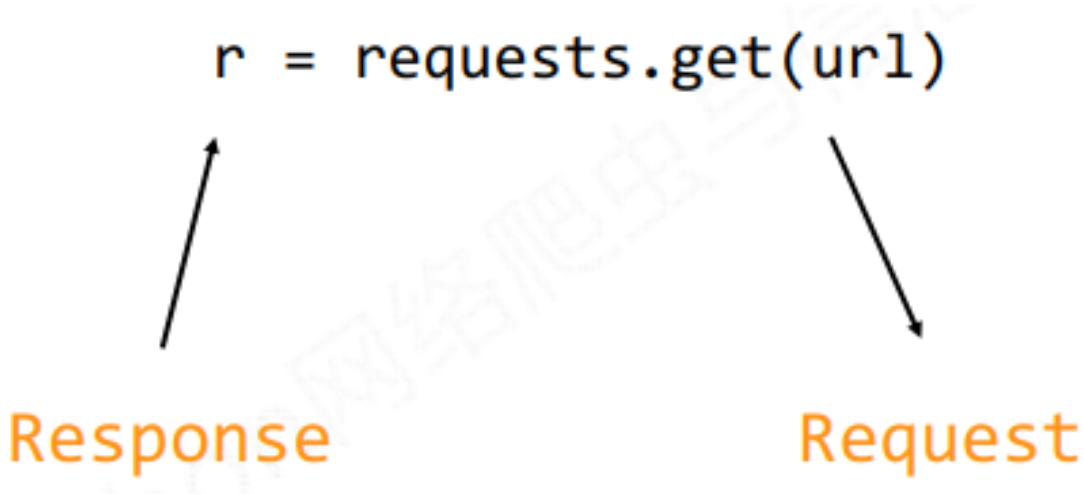
```
requests.get(url, params=None, **kwargs)
```

```python
def get(url, params=None, **kwargs):
    """Sends a GET request.

    :param url: URL for the new :class:`Request` object.
    :param params: (optional) Dictionary or bytes to be sent in the query string for the
    :param \*\*kwargs: Optional arguments that ``request`` takes.
    :return: :class:`Response <Response>` object
    :rtype: requests.Response
    """

    kwargs.setdefault('allow_redirects', True)
    return request('get', url, params=params, **kwargs)
```

request and response



Response contains the value that a web crawler gets.

Attributes of response

attributes	note
r.status_code	http response status, 200 means success, others means failure.
r.text	http response content, string form.
r.encoding	encoding of the http header.
r.apparent_encoding	encoding of the context.
r.content	http response content, binary form.

Encoding and apparent_encoding

```
>>> r = requests.get("http://www.baidu.com")
>>> r.status_code
200
>>> r.text
'<!DOCTYPE html>\r\n<!--STATUS OK--><html> <head><meta http-equiv=content-type c
ontent=text/html;charset=utf-8><meta http-equiv=X-UA-Compatible content=IE=Edge>
<meta content=always name=referrer><link rel=stylesheet type=text/css href=http:
//s1.bdstatic.com/r/www/cache/bdorz/baidu.min.css><title>ç\x99¾å°¦ä¸\x80ä¸\x8bï¼
>>> r.encoding
'ISO-8859-1'
>>> r.apparent_encoding
'utf-8'
>>> r.encoding = "utf-8"
>>> r.text
'<!DOCTYPE html>\r\n<!--STATUS OK--><html> <head><meta http-equiv=content-type c
ontent=text/html;charset=utf-8><meta http-equiv=X-UA-Compatible content=IE=Edge>
<meta content=always name=referrer><link rel=stylesheet type=text/css href=http:
//s1.bdstatic.com/r/www/cache/bdorz/baidu.min.css><title>百度一下，你就知道</titl
```

Handle the error

```
import requests

def getHTML(url):
    try:
        r = requests.get(url)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return "error"
```

Example 1 :requests.get(url, params)

When we create a request to https://httpbin.org/get

```
>>> import requests
>>> r = requests.get("https://httpbin.org/get")
>>> r.status_code
200
>>> print(r.text)
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.21.0"
  },
  "origin": "116.31.95.56, 116.31.95.56",
  "url": "https://httpbin.org/get"
}
```

Example 1 :requests.get(url, params)

When we need to add some extra information.

The URL could be something like http://httpbin.org/get?name=Tom&age=19 ??

We can use params!

```
>>> import requests
>>> data = {"name": "Tom", "age": "19"}
>>> r = requests.get("https://httpbin.org/get", params=data)
>>> r.status_code
200
>>> print(r.text)
{
  "args": {
    "age": "19",
    "name": "Tom"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.21.0"
  },
  "origin": "116.31.95.56, 116.31.95.56",
  "url": "https://httpbin.org/get?name=Tom&age=19"
}
```

Example 2: add user-agent

When we want to request https://zhihu.com/explore , we get a error like that:

```
>>> r = requests.get("https://zhihu.com/explore")
>>> r.status_code
400
>>> r.text
'<html>\r\n<head><title>400 Bad Request</title></head>\r\n<b
y</center>\r\n</body>\r\n</html>\r\n'
>>> r.request.headers
{'User-Agent': 'python-requests/2.21.0', 'Accept-Encoding':
```

Please have a look at the “User-Agent”, our program is very honest. :P

Example 2: add user-agent

So how to change the “User-Agent” so that the server cannot recognize this program?

```
>>> headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.3
6"}
>>> r = requests.get("https://zhihu.com/explore", headers = headers)
>>> r.status_code
200
>>> r.request.headers
{'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36', 'Accept-En
coding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive', '
Cookie': 'tgw_l7_route=a37704a413efa26cf3f23813004f1a3b'}
```

Example3: get picture

We want to get the github favicon: https://github.com/favicon.ico

```
>>> import requests
>>> url = "https://github.com/favicon.ico"
>>> r = requests.get(url)
>>> r.status_code
200
>>> path = "C://Users//Administrator//Desktop//favicon.ico"
>>> with open(path, "wb") as f:
...     f.write(r.content)
...
6518
```

Beautiful Soup

https://www.crummy.com/software/BeautifulSoup/

You didn't write that awful page. You're just trying to get some data out of it. Beautiful Soup is here to help. Since 2004, it's been saving programmers hours or days of work on quick-turnaround screen scraping projects.

Beautiful Soup

"A tremendous boon." -- Python411 Podcast

[Download | Documentation | Hall of Fame | Source | Changelog | Discussion group | Zine]

If you use Beautiful Soup as part of your work, please consider a Tidelift subscription. This will support many of the free software projects your organization depends on, not just Beautiful Soup.

If Beautiful Soup is useful to you on a personal level, you might like to read *Tool Safety*, a short zine I wrote about what I learned about software development from working on Beautiful Soup. Thanks!

If you have questions, send them to the discussion group. If you find a bug, file it.

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

1. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application
2. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then you just have to specify the original encoding.
3. Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

Beautiful Soup

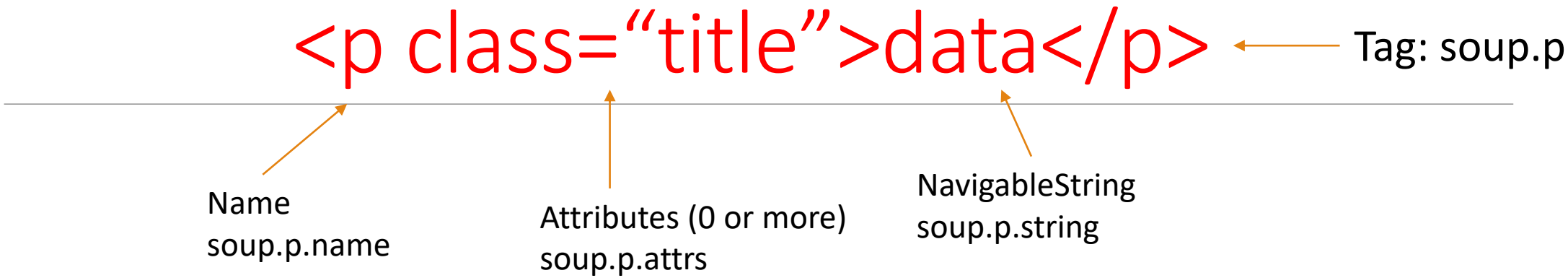
- How to install it?

  pip3 install beautifulsoup4

- Test code:

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup('<p>data</p>',"html.parser")
>>> print(soup.prettify())
<p>
 data
</p>
```

Beautiful Soup class basic elements



Basic elements	Notes
Tag	starts with <> and ends with </>. Usage: soup.<tag>
Name	… 's name is "p". Usage: <tag>.name
Attributes	Dictionary format. Usage: <tag> .attrs
NavigableString	e.g. data .string will return "data"
Comment	Comment in the tag

Resolver(解析器)

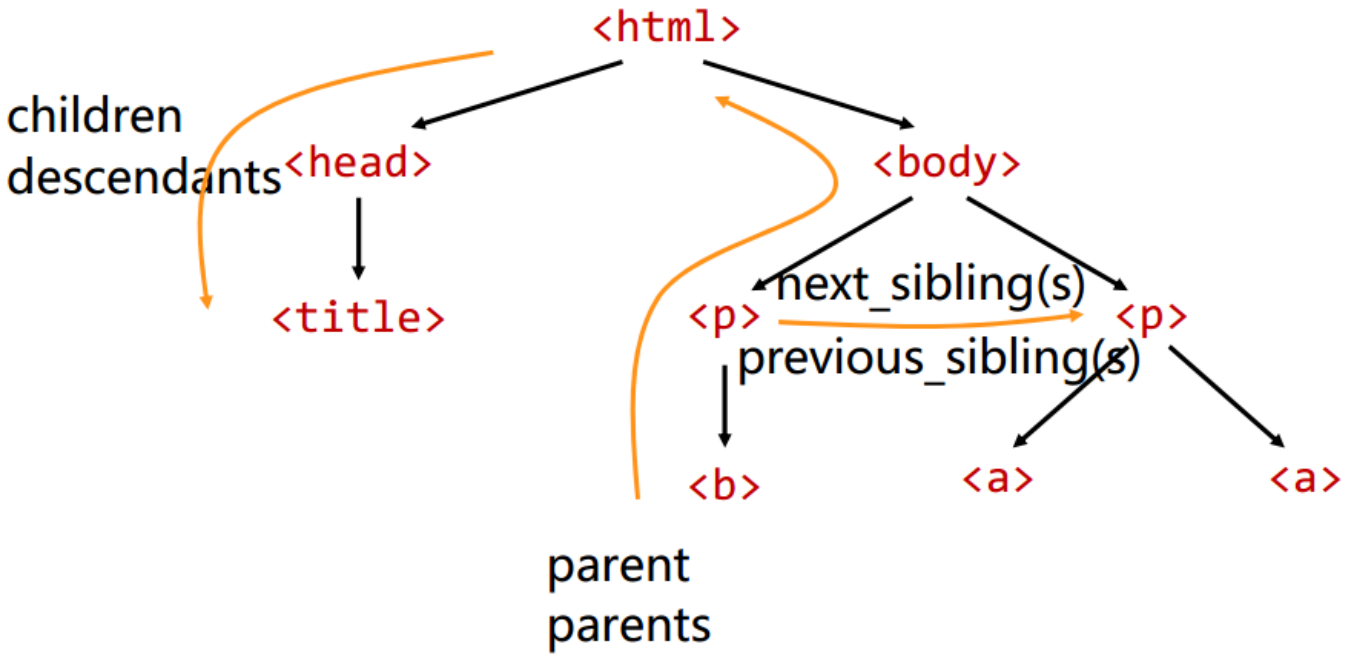
表 4-3　Beautiful Soup 支持的解析器

解　析　器	使用方法	优　　势	劣　　势
Python 标准库	BeautifulSoup(markup, "html.parser")	Python 的内置标准库、执行速度适中、文档容错能力强	Python 2.7.3 及 Python 3.2.2 之前的版本文档容错能力差
lxml HTML 解析器	BeautifulSoup(markup, "lxml")	速度快、文档容错能力强	需要安装 C 语言库
lxml XML 解析器	BeautifulSoup(markup, "xml")	速度快、唯一支持 XML 的解析器	需要安装 C 语言库
html5lib	BeautifulSoup(markup, "html5lib")	最好的容错性、以浏览器的方式解析文档、生成 HTML5 格式的文档	速度慢、不依赖外部扩展

Iterate through soup

We notice that soup.<Tag> will only return the first one. This is not good for search.

There are three ways to iterate through soup.



Children and descendants

Note that the result is a generator, which need you to use for loop. Example:

```
>>> import requests
>>> from bs4 import BeautifulSoup
>>> r = requests.get("https://www.baidu.com")
>>> r.status_code
200
>>> r.encoding = "UTF-8"
>>> soup = BeautifulSoup(r.text, "html.parser")
>>> for tag in soup.body.descendants:
...     print(tag.name)
...
None
div
None
div
None
div
```

Note that there are lots of “None”, we usually add some lines:

```
>>> import bs4
>>> for tag in soup.body.descendants:
...     if isinstance(tag, bs4.element.Tag):
...         print(tag.name)
```

find_all(name, attrs, recursive, text, **kwargs)

- Name: tag.name. e.g. soup.find_all(name="a")
- Attrs: tag.attrs. e.g. soup.find_all(attrs = {"class"="title"}).

  simpler way: soup.find_all(class_ = "title") or: soup(class_ = "title")
- Recursive: Boolean value. Recursive means only search in the children of current tag.
- Text: soup.find_all(text = "text") returns a list of string. Usually use re here.

Example： douban top movies

url = https://movie.douban.com/top250?start=0&filter=

The output should be:

```
$ python test.py
No. 1 is: 肖申克的救赎
No. 2 is: 霸王别姬
No. 3 is: 这个杀手不太冷
No. 4 is: 阿甘正传
No. 5 is: 美丽人生
No. 6 is: 泰坦尼克号
No. 7 is: 千与千寻
No. 8 is: 辛德勒的名单
No. 9 is: 盗梦空间
No. 10 is: 忠犬八公的故事
No. 11 is: 机器人总动员
No. 12 is: 三傻大闹宝莱坞
No. 13 is: 海上钢琴师
No. 14 is: 放牛班的春天
No. 15 is: 楚门的世界
No. 16 is: 大话西游之大圣娶亲
No. 17 is: 星际穿越
No. 18 is: 龙猫
No. 19 is: 教父
No. 20 is: 熔炉
No. 21 is: 无间道
No. 22 is: 疯狂动物城
No. 23 is: 当幸福来敲门
No. 24 is: 怦然心动
No. 25 is: 触不可及
```

```python
import requests
from bs4 import BeautifulSoup
import bs4
import re

def getHTML(url):
    try:
        r = requests.get(url)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return ''

def getInfo(text):
    title = []
    soup = BeautifulSoup(text,"html.parser")
    for a in soup.find_all(name="a",href=re.compile("https://movie.douban.com/subject/")):
        if isinstance(a.span,bs4.element.Tag):
            title.append(a.span.string)
    return title

def show(title):
    for i, t in enumerate(title):
        print("No.",i+1,"is:",t)

def main():
    url = "https://movie.douban.com/top250?start=0&filter="
    HTML = getHTML(url)
    if HTML=="":
        print("cannot get HTML")
        return
    info = getInfo(HTML)
    show(info)

if __name__ == "__main__":
    main()
```

Where to go?

1. Beautiful Soup doc: https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/

2. Requests doc: http://docs.python-requests.org/zh_CN/latest/index.html

3. Python 3网络爬虫开发实战 ,崔庆才著 (highly recommended)

Download pdf at: https://pan.baidu.com/s/1oRFxCa-a1kLux1eHOdh_oQ

Thank you!