



general speaking: 那些库:

requests

逻辑: ① `r = requests.get(url)` ← 得到的 r 是 response 对象, 包含所有信息.
② `r.status_code` (检测爬取成功与否)

↓
`200` ↓
`404或其它 (wrong)`

③ 用 `r.text` 及其他方法操作

requests 库的八大方法对应 HTTP 的八种操作

13个参数

本章最重要的是 `request.head` 和 `request.get` 的运用
以及基本代理框架

Requests库的7个主要方法

方法	说明
requests.request()	构造一个请求，支撑以下各方法的基础方法
requests.get()	获取HTML网页的主要方法，对应于HTTP的GET
requests.head()	获取HTML网页头信息的方法，对应于HTTP的HEAD
requests.post()	向HTML网页提交POST请求的方法，对应于HTTP的POST
requests.put()	向HTML网页提交PUT请求的方法，对应于HTTP的PUT
requests.patch()	向HTML网页提交局部修改请求，对应于HTTP的PATCH
requests.delete()	向HTML页面提交删除请求，对应于HTTP的DELETE

属性	说明
r.status_code	HTTP请求的返回状态，200表示连接成功，404表示失败
r.text	HTTP响应内容的字符串形式，即，url对应的页面内容
r.encoding	从HTTP header中猜测的响应内容编码方式
r.apparent_encoding	从内容中分析出的响应内容编码方式（备选编码方式）
r.content	HTTP响应内容的二进制形式

Eg. 1:

```
import requests
r = requests.get("http://www.google.com")
print(r.status_code) —— 200 is good!
r.encoding = "UTF-8"
print(r.text)
```

异常处理

异常	说明
requests.ConnectionError	网络连接错误异常，如DNS查询失败、拒绝连接等
requests.HTTPError	HTTP错误异常
requests.URLRequired	URL缺失异常
requests.TooManyRedirects	超过最大重定向次数，产生重定向异常
requests.ConnectTimeout	连接远程服务器超时异常
requests.Timeout	请求URL超时，产生超时异常

异常	说明
r.raise_for_status()	如果不是200，产生异常requests.HTTPError

爬虫结构

爬取网页的通用代码框架

```

import requests

def getHTMLText(url):
    try:
        r = requests.get(url, timeout=30)
        r.raise_for_status() #如果状态不是200, 引发HTTPError异常
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return "产生异常"

if __name__ == "__main__":
    url = "http://www.baidu.com"
    print(getHTMLText(url))

```

HTTP协议与request的方法

方法	说明
GET	请求获取URL位置的资源
HEAD	请求获取URL位置资源的响应消息报告, 即获得该资源的头部信息
POST	请求向URL位置的资源后附加新的数据
PUT	请求向URL位置存储一个资源, 覆盖原URL位置的资源
PATCH	请求局部更新URL位置的资源, 即改变该处资源的部分内容
DELETE	请求删除URL位置存储的资源



理解PATCH和PUT的区别

假设URL位置有一组数据UserInfo，包括UserID、UserName等20个字段。

需求：用户修改了UserName，其他不变。

- 采用PATCH，仅向URL提交UserName的局部更新请求。
- 采用PUT，必须将所有20个字段一并提交到URL，未提交字段被删除。

head()方法：

Requests库的head()方法

```

>>> r = requests.head('http://httpbin.org/get')
>>> r.headers
{'Content-Length': '238', 'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Credentials': 'true', 'Content-Type': 'application/json', 'Server': 'nginx', 'Connection': 'keep-alive', 'Date': 'Sat, 18 Feb 2017 12:07:44 GMT'}
>>> r.text

```

资源概要

post方法：

Requests库的post()方法

```

>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.post('http://httpbin.org/post', data = payload)
>>> print(r.text)
{
  ...
  "form": {
    "key2": "value2",
    "key1": "value1"
  }
}

```

向URL POST一个字典
自动编码为form (表单)

put方法

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.put("http://httpbin.org/put", data = payload)
>>> print(r.text)
{
  ...
  "form": {
    "key2": "value2",
    "key1": "value1"
  }
}
```

request方法解析:

requests.request(method, url, **kwargs)

method: 请求方式, 对应get/put/post等7种
url: 拟获取页面的url链接
****kwargs**: 控制访问的参数, 共13个

```
requests.request(method, url, **kwargs)
method: 请求方式
r = requests.request('GET', url, **kwargs)
r = requests.request('HEAD', url, **kwargs)
r = requests.request('POST', url, **kwargs)
r = requests.request('PUT', url, **kwargs)
r = requests.request('PATCH', url, **kwargs)
r = requests.request('DELETE', url, **kwargs)
r = requests.request('OPTIONS', url, **kwargs)
```

相关参数

requests.request(method, url, **kwargs)

****kwargs**: 控制访问的参数, 均为可选项
params: 字典或字节序列, 作为参数增加到url中

```
>>> kv = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.request('GET', 'http://python123.io/ws', params=kv)
>>> print(r.url)
http://python123.io/ws?key1=value1&key2=value2
```

requests.request(method, url, **kwargs)

****kwargs**: 控制访问的参数, 均为可选项
data: 字典、字节序列或文件对象, 作为Request的内容

```
>>> kv = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.request('POST', 'http://python123.io/ws', data=kv)
>>> body = r.text
>>> r = requests.request('POST', 'http://python123.io/ws', data=body)
```

requests.request(method, url, **kwargs)

****kwargs**: 控制访问的参数, 均为可选项
proxies: 字典类型, 设定访问代理服务器, 可以增加登录认证

```
>>> pxs = { 'http': 'http://user:pass@10.10.10.1:1234'
           'https': 'https://10.10.10.1:4321' }
>>> r = requests.request('GET', 'http://www.baidu.com', proxies=pxs)
```

requests.request(method, url, **kwargs)

****kwargs**: 控制访问的参数, 均为可选项
json: JSON格式的数据, 作为Request的内容

```
>>> kv = {'key1': 'value1'}
>>> r = requests.request('POST', 'http://python123.io/ws', json=kv)
```

requests.request(method, url, **kwargs)

****kwargs**: 控制访问的参数, 均为可选项
allow_redirects: True/False, 默认为True, 重定向开关
stream: True/False, 默认为True, 获取内容立即下载开关
verify: True/False, 默认为True, 认证SSL证书开关
cert: 本地SSL证书路径

requests.request(method, url, **kwargs)

****kwargs**: 控制访问的参数, 均为可选项
headers: 字典, HTTP定制头

```
>>> hd = {'user-agent': 'Chrome/10'}
>>> r = requests.request('POST', 'http://python123.io/ws', headers=hd)
```

requests.request(method, url, **kwargs)

****kwargs**: 控制访问的参数, 均为可选项
cookies: 字典或CookieJar, Request中的cookie
auth: 元组, 支持HTTP认证功能

files: 字典类型, 传输文件

```
>>> fs = {'file': open('data.xls', 'rb')}
>>> r = requests.request('POST', 'http://python123.io/ws', files=fs)
```

一些技巧：①技术手段限制
 ②robots 协议：①不遵守有法律风险
 ②类人行为不道德

具体爬虫实例

Amazon 爬虫实例：更改 user-agent 信息。
 ↳ 关于其破的搞笑故事

对读取内容的解析：beautiful soup 序：

几种解析器：

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

Demo:



beautifulsoup 基本元素：

基本元素	说明
Tag	标签，最基本的信息组织单元，分别用<>和</>标明开头和结尾
Name	标签的名字，<p>...</p>的名字是'p'，格式：<tag>.name
Attributes	标签的属性，字典形式组织，格式：<tag>.attrs
NavigableString	标签内非属性字符串，<>...</>中字符串，格式：<tag>.string
Comment	标签内字符串的注释部分，一种特殊的Comment类型

获取标签内容：

```
1 import requests
2 from bs4 import BeautifulSoup
3 r = requests.get("https://python123.io/ws/demo.html")
4 demo = r.text
5 soup = BeautifulSoup(demo, "html.parser")
6 print(soup.prettify())
7 ### 获取标签内容
8 print(soup.title)
9 print(soup.p)
```

输出第一个p标签内容

获取标签名字

```
### 获取标签名字
print(soup.a.name)
print(soup.a.parent.name)
print(soup.a.parent.parent.name)
print(soup.a.parent.parent.parent.name)
```

requests.request(method, url, **kwargs)

**kwargs：控制访问的参数，均为可选项

params	cookies	proxies	cert
data	auth	allow_redirects	
json	files	stream	
headers	timeout	verify	

获取标签属性

```
### 获取标签属性
tag = soup.a
print(type(tag.attrs))
print(tag.attrs["class"])
print(tag.attrs["id"])
print(tag.attrs["href"])
```

NavigableString

```
25
26 #####navigable string
27
28 print(soup.p.string)
29 print(type(soup.p.string))
```

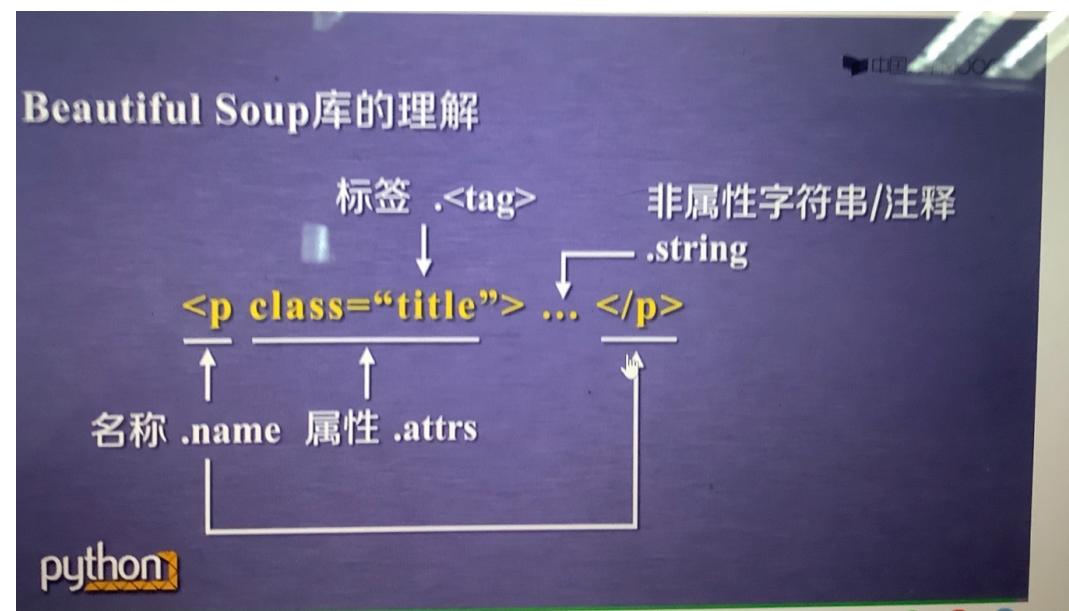
The demo python introduces several python courses.
 <class 'bs4.element.NavigableString'>

获取注释 (comment)

```

Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>> newsoup = BeautifulSoup("<b><!--This is a comment--></b><p>This is not a comment</p>", "html.parser")
This is not a comment
>>> newsoup.b.string
'This is a comment'
>>> type(newsoup.b.string)
<class 'bs4.element.Comment'>
>>> newsoup.p.string
'This is not a comment'
>>> type(newsoup.p.string)
<class 'bs4.element.NavigableString'>

```



遍历结构 ① 下行遍历：



标签树的下行遍历

属性	说明
.contents	子节点的列表，将<tag>所有儿子节点存入列表
.children	子节点的迭代类型，与.contents类似，用于循环遍历儿子节点
.descendants	子孙节点的迭代类型，包含所有子孙节点，用于循环遍历

```

Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>> soup = BeautifulSoup(demo, "html.parser")
>>> soup.head
<head><title>This is a python demo page</title></head>
>>> soup.head.contents
[<title>This is a python demo page</title>]
>>> soup.body.contents
['\n', <p class="title"><b>The demo python introduces several p
ython courses.</b></p>, '\n', <p class="course">Python is a won
derful general-purpose programming language. You can learn Pyth
on from novice to professional by tracking the following course
s:
<a class="py1" href="http://www.icourse163.org/course/BIT-26800
1" id="link1">Basic Python</a> and <a class="py2" href="http://
www.icourse163.org/course/BIT-1001870001" id="link2">Advanced P
ython</a>.</p>, '\n']
>>> len(soup.body.contents)
5
>>> |
>>> soup.body.contents[1]
<p class="title"><b>The demo python introduces several python c
ourses.</b></p>

```

```

##content##
print(tag1.contents)
print(tag2.contents)

##children##
for child in tag2.children:
    print [child,end=""]

```

② 标签树的上行遍历

属性	说明
.parent	节点的父亲标签
.parents	节点先辈标签的迭代类型，用于循环遍历先辈节点

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>> soup = BeautifulSoup(demo, "html.parser")
>>> soup.title.parent
<head><title>This is a python demo page</title></head>
>>> soup.html.parent
<html><head><title>This is a python demo page</title></head>
<body>
<p class="title"><b>The demo python introduces several python c
ourses.</b></p>
<p class="course">Python is a wonderful general-purpose program
ming language. You can learn Python from novice to professional
by tracking the following courses:
<a class="py1" href="http://www.icourse163.org/course/BIT-26800
1" id="link1">Basic Python</a> and <a class="py2" href="http://
www.icourse163.org/course/BIT-1001870001" id="link2">Advanced P
ython</a>.</p>
</body></html>
```

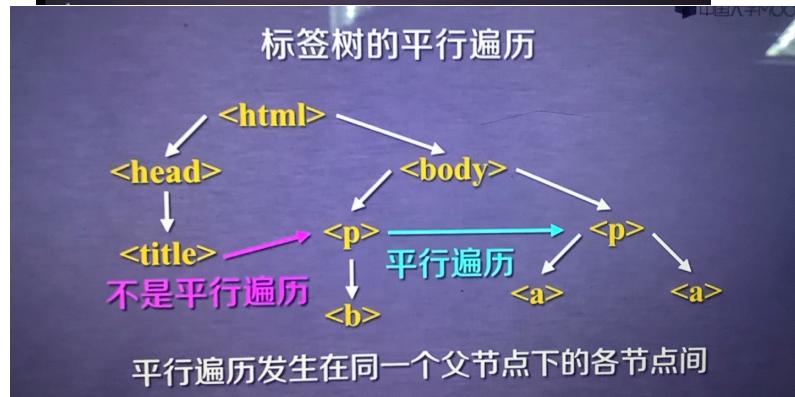
标签树的上行遍历

```
>>> soup = BeautifulSoup(demo, "html.parser")
>>> for parent in soup.a.parents:
...     if parent is None:
...         print(parent)
...     else:
...         print(parent.name)

p
body
html
[document]
```

③ 平行遍历

属性	说明
.next_sibling	返回按照HTML文本顺序的下一个平行节点标签
.previous_sibling	返回按照HTML文本顺序的上一个平行节点标签
.next_siblings	迭代类型，返回按照HTML文本顺序的后续所有平行节点标签
.previous_siblings	迭代类型，返回按照HTML文本顺序的前续所有平行节点标签



```
>>> soup = BeautifulSoup(demo, "html.parser")
>>> soup.a.next_sibling
' and '
>>> soup.a.next_sibling.next_sibling
<a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Advanced Python</a>
```

如何让HTML格式更好显示：prettyify方法

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(demo, "html.parser")
>>> soup.prettify()
<html>
<head>
<title> This is a python demo page</title>
</head>
<body>
<p class="title"> This is a python demo page</p>
</body>
</html>
```

bs4库的prettyify()方法

```
>>> print(soup.a.prettify())
<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">
Basic Python
</a>
```

信息标记

XML:

XML eXtensible Markup Language

```
<name> ... </name>
<name />
<!-- -->
```

XML eXtensible Markup Language

标签 Tag

```
 ... </img>
```

名称 Name 属性 Attribute

JSON:

JSON JavaScript Object Notation

```
"key" : "value"
"key" : ["value1", "value2"]
"key" : {"subkey" : "subvalue"}
```

YAML:

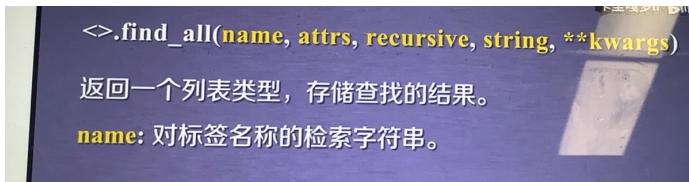
YAML Ain't Markup Language

```
name :
  newName : 北京理工大学
  oldName : 延安自然科学院
```

YAML Ain't Markup Language

```
key : value
key : #Comment → 注释
  -value1
  -value2
key :
  subkey : subvalue
```

soup 的信息处理函数



<tag>(..) 等价于 <tag>.find_all(..)

soup(..) 等价于 soup.find_all(..)

方法	说明
<>.find()	搜索且只返回一个结果，字符串类型，同.find_all()参数
<>.find_parents()	在先辈节点中搜索，返回列表类型，同.find_all()参数
<>.find_parent()	在先辈节点中返回一个结果，字符串类型，同.find()参数
<>.find_next_siblings()	在后续平行节点中搜索，返回列表类型，同.find_all()参数
<>.find_next_sibling()	在后续平行节点中返回一个结果，字符串类型，同.find()参数
<>.find_previous_siblings()	在前序平行节点中搜索，返回列表类型，同.find_all()参数
<>.find_previous_sibling()	在前序平行节点中返回一个结果，字符串类型，同.find()参数

python

<td> 4td>
<td> 北京 <p>cp> <td>
<td> 北京 <td>
<td> 4字节 <td>
<td> 832.5 <td>

正则表达式

正则表达式的常用操作符

操作符	说明	实例
.	表示任何单个字符	
[]	字符集, 对单个字符给出取值范围	[abc]表示a、b、c, [a-z]表示a到z单个字符
[^]	非字符集, 对单个字符给出排除范围	[^abc]表示非a或b或c的单个字符
*	前一个字符0次或无限次扩展	abc* 表示 ab、abc、abcc、abccc等
+	前一个字符1次或无限次扩展	abc+ 表示 abc、abcc、abccc等
?	前一个字符0次或1次扩展	abc? 表示 ab、abc
	左右表达式任意一个	abc def 表示 abc、def

正则表达式的常用操作符

操作符	说明	实例
{m}	扩展前一个字符m次	ab{2}c表示abbc
{m,n}	扩展前一个字符m至n次(含n)	ab{1,2}c表示abc、abbc
^	匹配字符串开头	^abc表示abc且在一个字符串的开头
\$	匹配字符串结尾	abc\$表示abc且在一个字符串的结尾
()	分组标记, 内部只能使用 操作符	(abc)表示abc, (abc def)表示abc、def
\d	数字, 等价于[0-9]	
\w	单词字符, 等价于[A-Za-z0-9_]	

正则表达式

对应字符串

P(Y YT YTH YTHO)?N	'PN'、'PYN'、'PYTN'、'PYTHN'、'PYTHON'
PYTHON+	'PYTHON'、'PYTHONN'、'PYTHONNN' ...
PY[TH]ON	'PYTON'、'PYHON'
PY[^TH]?ON	'PYON'、'PYaON'、'PYbON'、'PYcON' ...
PY{:3}N	'PN'、'PYN'、'PYYN'、'PYYYN'

^[A-Za-z]+\$	由26个字母组成的字符串
^[A-Za-z0-9]+\$	由26个字母和数字组成的字符串
^-?\d+\$	整数形式的字符串
^[-?]\d+\$	正整数形式的字符串
^[-?]\d{5}\$	中国境内邮政编码, 6位
[\u4e00-\u9fa5]	匹配中文字符
\d{3}-\d{8} \d{4}-\d{7}	国内电话号码, 010-68913536

匹配IP地址的正则表达式

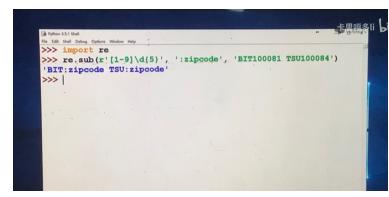
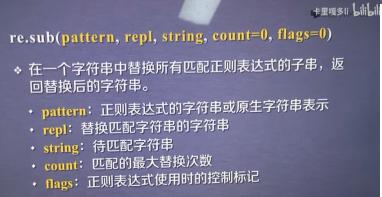
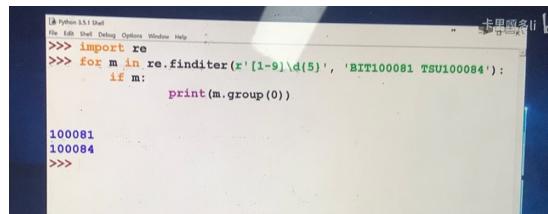
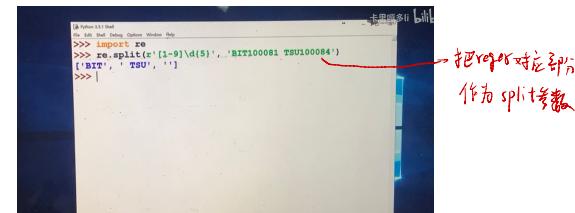
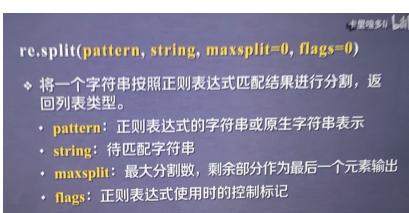
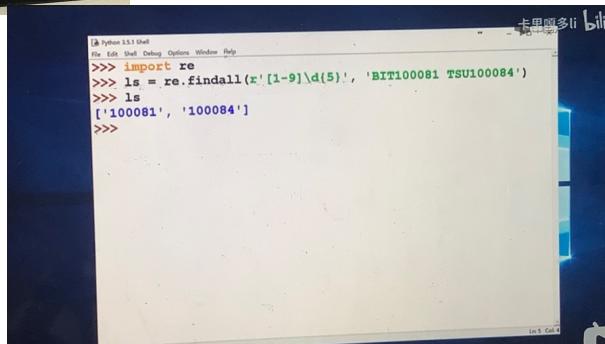
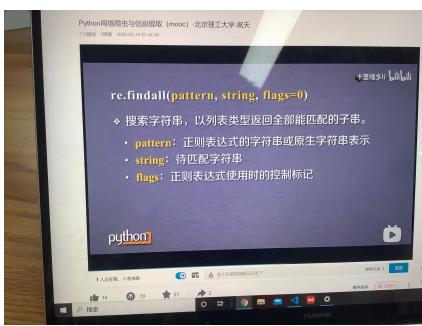
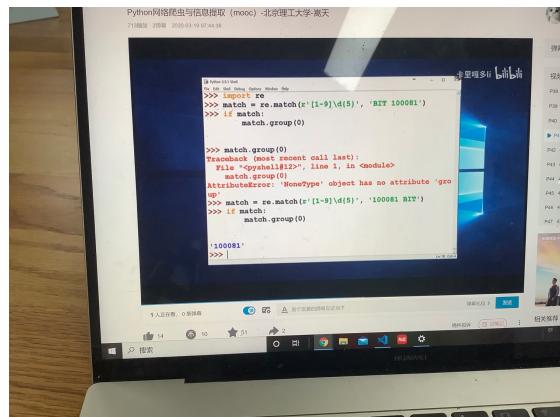
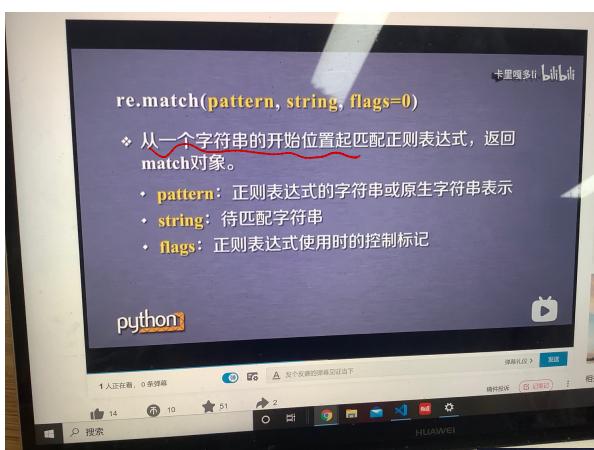
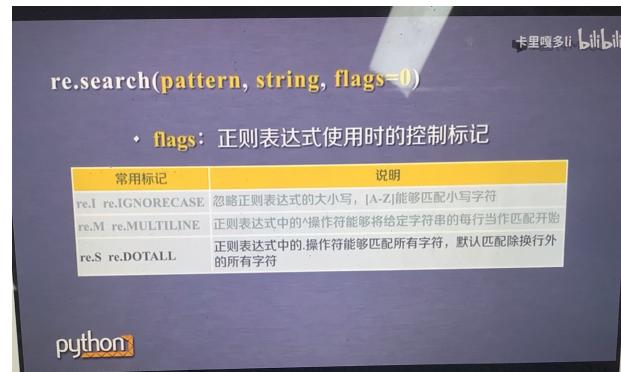
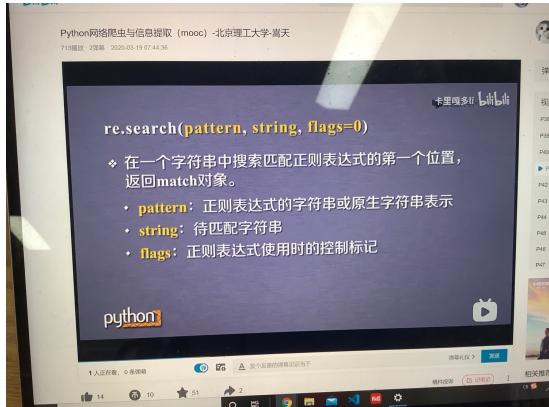
IP地址字符串形式的正则表达式
(IP地址分4段, 每段0-255)

精确写法

0-99: [1-9]? \d	100-199: 1 \d{2}
200-249: 2 [0-4] \d	250-255: 25 [0-5]
(([1-9]? \d 1 \d{2} 2 [0-4] \d 25 [0-5]) .) {3} (([1-9]? \d 1 \d{2} 2 [0-4] \d 25 [0-5])	

规律的兀立

函数	说明
re.search()	在一个字符串中搜索匹配正则表达式的第一位置，返回match对象
re.match()	从一个字符串的开始位置起匹配正则表达式，返回match对象
re.findall()	搜索字符串，以列表类型返回全部能匹配的子串
re.split()	将一个字符串按照正则表达式匹配结果进行分割，返回列表类型
re.finditer()	搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象
re.sub()	在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串



另一种正则表达式的用法

编译匹配

```
regex = re.compile(pattern, flags=0)
将正则表达式字符串编译成正则表达式对象
• pattern: 正则表达式的字符串或原生字符串表示
• flags: 正则表达式使用时的控制标记
>>> regex = re.compile(r'[1-9]\w{5}')
```

Re库的另一种等价用法	
函数	说明
regex.search()	在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象
regex.match()	从一个字符串的开始位置匹配正则表达式，返回match对象
regex.findall()	搜索字符串，以列表类型返回全部能匹配的子串
regex.split()	将一个字符串按照正则表达式匹配结果进行分割，返回列表类型
regex.finditer()	搜索字符串，返回一个匹配结果的迭代器，每个迭代元素是match对象
regex.sub()	在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

Match 对象

```
Python网络爬虫与信息提取 (mooc) - 北京理工大学-嵩天
113播放 29弹幕 2020-03-19 07:44:36

>>> match = re.search(r'[1-9]\d{5}', 'BIT 100081')
>>> if match:
...     print(match.group(0))

100081
>>> type(match)
<class '_sre.SRE_Match'>
>>> |
```

Match对象的属性	
属性	说明
string	待匹配的文字
re	匹配时使用的pattern对象（正则表达式）
pos	正则表达式搜索文字的开始位置
endpos	正则表达式搜索文字的结束位置

Match对象的方法

方法	说明
.group(0)	获得匹配后的字符串
.start()	匹配字符串在原始字符串的开始位置
.end()	匹配字符串在原始字符串的结束位置
.span()	返回(.start(), .end())

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>> import re
>>> m = re.search(r'[1-9]\d{5}', 'BIT100081 TSU100084')
'BIT100081 TSU100084'
>>> m.re
re.compile('[1-9]\d{5}')
>>> m.pos
0
>>> m.endpos
19
>>> m.group(0)
'100081'
>>> m.start()
3
>>> m.end()
9
>>> m.span()
(3, 9)
>>> |
```

贪婪匹配 (最长匹配) V.S. 非贪婪匹配

实例

```
>>> match = re.search(r'PY.*N', 'PYANBNCON')
>>> match.group(0)
```

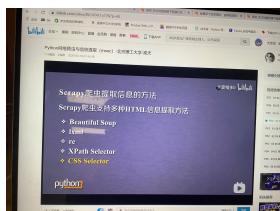
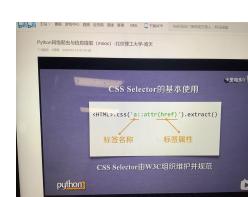
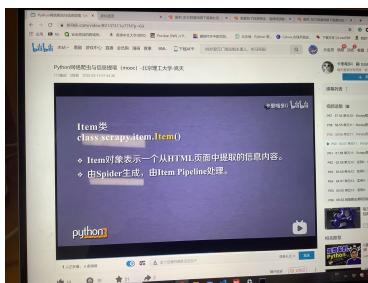
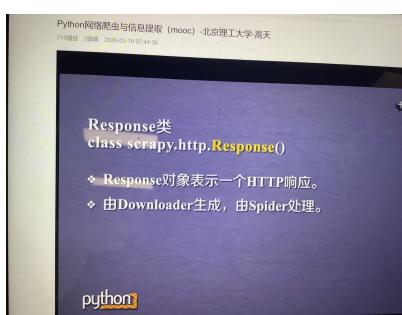
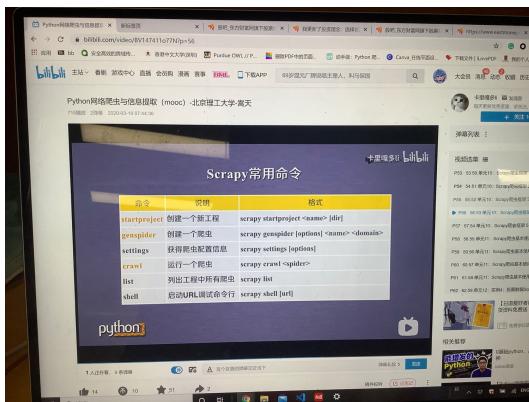
默认贪婪匹配。

最小匹配操作符 → 最小匹配方法	
操作符	说明
*?	前一个字符0次或无限次扩展，最小匹配
+?	前一个字符1次或无限次扩展，最小匹配
??	前一个字符0次或1次扩展，最小匹配
{m,n}?	扩展前一个字符m至n次（含n），最小匹配

爬取股票数据网站选取原则:

- 候选数据网站的选择
- 选取原则：股票信息静态存在于HTML页面中，非js代码生成，没有Robots协议限制。
 - 选取方法：浏览器F12，源代码查看等。

Scrapy 爬虫框架



实例：股票爬虫实例



```
# -*- coding: utf-8 -*-
# scrapy
# import re

class StockSpider(scrapy.Spider):
    name = "stocks"
    start_urls = ['http://quote.eastmoney.com/stocklist.html']

    def parse(self, response):
        for href in response.css('a::attr(href)').extract():
            try:
                stock = re.findall(r'<a>(.*?)</a>', href)[0]
                url = 'http://quotation.eastmoney.com/stock/' + stock + '.html'
                yield scrapy.Request(url, callback=self.parse_stock)
            except:
                continue

    def parse_stock(self, response):
        infoList = []
        stockInfo = response.css('table').get()
        name = stockInfo.css('.bets-name').extract()[0]
        keyInfo = stockInfo.css('.keyinfo').extract()[0]
        valueList = stockInfo.css('.dd').extract()
        for i in range(len(valueList)):
            key = re.findall(r'<td>(.*?)</td>', keyInfo[i])[0][1:-1]
            val = re.findall(r'<td>(.*?)</td>', valueList[i])[0][0:-5]
            infoList.append({key: val})
        infoList.append({
            '股票名称': re.findall(r'<td>(.*?)</td>', name)[0].split()[0] + \
            re.findall(r'<td>(.*?)</td>', name)[0][1:-1]
        })
        yield infoList
```

实际上没用上面的结构。

 ↗ 阅读
<cite> 162 </cite>
<cite> 1 ↗ 引用
 新济医疗吧 <a> 新济医疗吧 股票代码 600887 股票内容 标题
 <input type="checkbox" />
<li class="date"> 11-21
<li class="last"> 11-21 15:17
