2025年春

程序设计实习：Python程序设计

第十六讲 网络爬虫设计

刘家瑛

liujiaying@pku.edu.cn

01 | 爬虫的用途和原理

爬虫的用途

- 在网络上搜集数据 (例如: 搜索引擎)
- 模拟浏览器快速操作 (抢票, 抢课, 抢挂号...)
- 模拟浏览器操作, 替代填表等重复操作

最基本的爬虫写法

数据获取型爬虫的本质就是自动获取网页并抽取其中的内容

① 手工找出合适的URL (网址)

② 用浏览器手工查看URL对应的网页, 并查看网页源码, 找出包含想要的内容 (文件名, 链接等) 的字符串的模式

③ 程序中获取URL对应的网页

④ 程序中用正则表达式或BeautifulSoup库或通过浏览器查找元素功能抽取网页中想要的内容并保存

示例：获取百度图片的搜索结果图片

1. 在百度图片敲关键字“desk”进行搜索

2. 搜索后看浏览器地址栏的地址：

https://image.baidu.com/search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&dyTabStr=MCwxMiwzLDEsMiwxMyw3LDYsNSw5&word=**desk**

示例：获取百度图片的搜索结果图片

3. 猜测只要在浏览器输入下面地址的红色部分，替换蓝色部分的单词，就能搜到图片：

https://image.baidu.com/search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8&dyTabStr=MCwxMiwzLDEsMiwxMyw3LDYsNSw5&word=猫

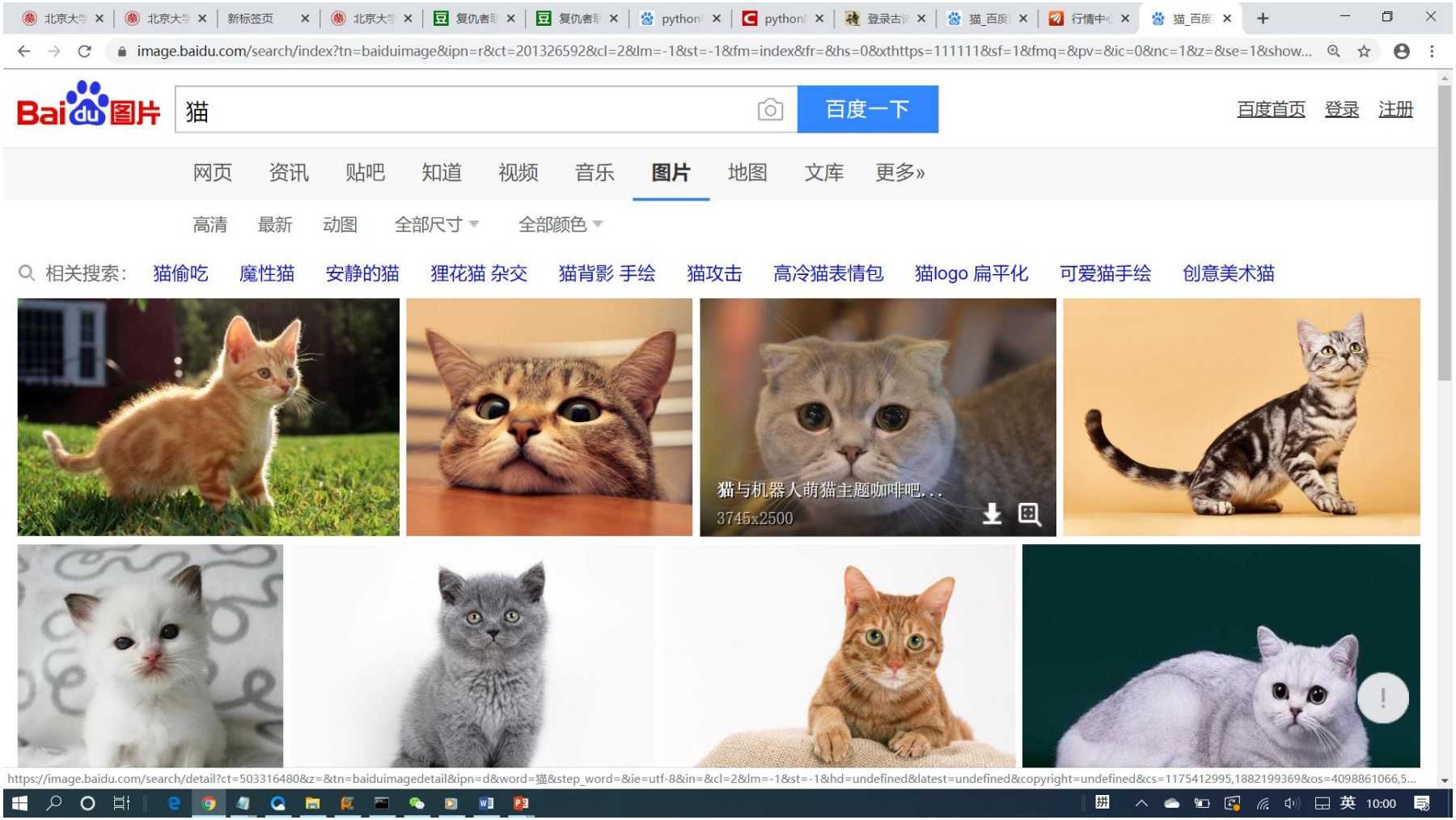


image.baidu.com/search/index?tn=baiduimage&ipn=r&ct=201326592&cl=2&lm=-1&st=-1&fm=index&fr=&hs=0&xthttps=111111&sf=1&fmq=&pv=&ic=0&nc=1&z=&se=1&show...
Bai du 图片
猫
百度一下
百度首页 登录 注册
网页 资讯 贴吧 知道 视频 音乐 图片 地图 文库 更多»
高清 最新 动图 全部尺寸 全部颜色
相关搜索： 猫偷吃 魔性猫 安静的猫 狸花猫 杂交 猫背影 手绘 猫攻击 高冷猫表情包 猫logo 扁平化 可爱猫手绘 创意美术猫
猫与机器人萌猫主题咖啡吧...
3745x2500
https://image.baidu.com/search/detail?ct=503316480&z=&tn=baiduimagedetail&ipn=d&word=猫&step_word=&ie=utf-8&in=&cl=2&lm=-1&st=-1&hd=undefined&latest=undefined&copyright=undefined&cs=1175412995,1882199369&os=4098861066,5...
英 10:00

鼠标右键点“下载”图标，在弹出的

菜单上点“复制链接”复制图片的URL

复制出来的图片地址：

https://img2.baidu.com/it/u=519022685,452596173&fm=253&fmt=auto&app=120&f=JPEG?w=800&h=800

此链接是百度保存的缩略图的网址

示例：获取百度图片的搜索结果图片

4. 在搜索结果界面，用浏览器查看源码

5. 在源码中查找

```
u=519022685,452596173&fm=253&fmt=auto&app=120&f=JPEG?w=800&h=800
```

找到：

\u0026 是 & 的Unicode转义序列：
- \u 表示Unicode转义
- 0026 是 & 的Unicode码点（十六进制）

…

```
{"thumburl":"https://img2.baidu.com/it/u=519022685,452596173\u0026fm=253\u0026fmt=auto\u0026app=120\u0026f=JPEG?w=800\u0026h=800",
```

…

可以用正则表达式提取图片链接

```
import re                #使用正则表达式
import requests          #request库用于获取网络资源 pip install requests
def getHtml(url):        #获取网址为url的网页,返回值是个字符串
#具体实现略,后面再讲
def getBaiduPictures(word, n):
#下载n个百度图片搜来的关于word的图片保存到本地
    url ="https://image.baidu.com/search/index?tn=baiduimage&ipn=r
    &ct=201326592&cl=2&lm=-1&st=1&fm=index&fr=&hs=0
    &xthttps=111111&sf=1&fmq=&pv=&ic=0&nc=1&z=&se=1&showtab=0&fb=0
    &width=&height=&face=0&istype=2&ie=utf-8&word="
    url += word
    html = getHtml(url)
    pt = '\"thumburl\":.*?\"(.*?)\"' #正则表达式,用于寻找图片网址
    i = 0
#"thumburl":"https://img2.baidu.com/it/u=519022685,452596173\u002
6fm=253\u0026fmt=auto\u0026app=120\u0026f=JPEG?w=800\u0026h=800",
```

```
    for x in re.findall(pt, html):                  #x就是图片url
        print(x)
        x = x.replace(r'\u0026', '&')               #额外处理转义字符
        try:
            r = requests.get(x, stream=True) #获取x对应的网络资源
            f = open('{0}{1}.jpg'.format(word, i), "wb")
            #"wb"表示二进制写方式打开文件
            f.write(r.content)                      #图片内容写入文件
            f.close()
            i = i + 1
        except Exception as e :
            pass
        if i >= n:
            break
getBaiduPictures("猫", 5)
getBaiduPictures("熊猫", 5)
```


02 | 用requests库获取网页

用requests.get获取网页

```
import requests
#requests库用于获取网络资源,pip install requests
```

requests库：Python中用于发送 HTTP 请求的第三方库，

需通过 pip install requests 安装

作用：发起 GET 请求并获取网页内容

用requests.get获取网页

```
def getHtml(url):  #获取网址url的网页
    import requests
    fakeHeaders = {'User-Agent':  #用于伪装浏览器发送请求
        'Mozilla/5.0 (Windows NT 10.0; Win64; x64)  \
        AppleWebKit/537.36 (KHTML, like Gecko) \
        Chrome/81.0.4044.138 Safari/537.36 Edg/81.0.416.77',
        'Accept': 'text/html, application/xhtml+xml, */*'}
    try:
        r = requests.get(url, headers = fakeHeaders)
        r.encoding = r.apparent_encoding #确保网页编码正确
        return r.text  #返回值是个字符串,内含整个网页内容
    except Exception as e:
        print(e)
        return ""
#用法: html = getHtml("http://openjudge.cn")
```


Http请求头 fakeHeaders

```
fakeHeaders = {'User-Agent':  #用于伪装浏览器发送请求
       'Mozilla/5.0 (Windows NT 10.0; Win64; x64)  \
       AppleWebKit/537.36 (KHTML, like Gecko) \
       Chrome/81.0.4044.138 Safari/537.36 Edg/81.0.416.77',
       'Accept': 'text/html,application/xhtml+xml,*/*'}
```

User-Agent

作用：告诉服务器你使用的浏览器和操作系统信息

含义：

Mozilla/5.0: 历史遗留字段，现代浏览器都会带这个标识

Windows NT 10.0; Win64; x64: 表示运行在 Windows 10 64位系统

AppleWebKit/537.36: 浏览器引擎(Chrome/Edge/Safari都使用它)

Chrome/81.0.4044.138: 基于Chromium的浏览器 (如Chrome或Edge)

Safari/537.36: 兼容 Safari 的渲染引擎

Edg/81.0.416.77: 表示这是 Microsoft Edge 浏览器(版本 81)

Http请求头 fakeHeaders

```
fakeHeaders = {'User-Agent':  #用于伪装浏览器发送请求
       'Mozilla/5.0 (Windows NT 10.0; Win64; x64)  \
       AppleWebKit/537.36 (KHTML, like Gecko) \
       Chrome/81.0.4044.138 Safari/537.36 Edg/81.0.416.77',
       'Accept': 'text/html,application/xhtml+xml,*/*'}
```

Agent

作用: 告诉服务器客户端可以接收哪些类型的响应数据

含义:

text/html: 优先接收 HTML 格式的数据。

application/xhtml+xml: 也可以接收 XHTML 格式

***/*:** 如果前两种不可用, 可以接受任何类型的数据

Http请求头 fakeHeaders

```
fakeHeaders = {'User-Agent':   #用于伪装浏览器发送请求
       'Mozilla/5.0 (Windows NT 10.0; Win64; x64)   \
       AppleWebKit/537.36 (KHTML, like Gecko) \
       Chrome/81.0.4044.138 Safari/537.36 Edg/81.0.416.77',
       'Accept': 'text/html,application/xhtml+xml,*/*`}
```

Agent

作用： 告诉服务器客户端可

含义：

text/html: 优先接收 HT

application/xhtml+xm

***/*:** 如果前两种不可用，任

为什么需要 fakeHeaders?

1. **避免被反爬虫拦截**
   - 某些网站会检查**User-Agent**，如果发现是Python的默认UA，可能会拒绝响应或返回错误数据
   - 通过伪装成浏览器，可以绕过简单的反爬机制
2. **模拟真实用户访问**
   - 有些网站会根据**Accept**返回不同的内容（如PC版或移动版页面），设置合理的**Accept**可以确保获取正确的数据
3. **兼容性**
   - 部分网站依赖**User-Agent**判断浏览器类型，如果没有正确的UA，可能导致JavaScript/CSS加载异常

用requests库获取网页的优势和局限:

优势

- 相比其它方法, 速度快几倍
- 安装简单, 分发容易

局限

- 容易被反爬虫手段屏蔽
- 不能获取包含javascript生成的动态网页

弥补requests不足的其它获取网页的办法

- 使用**selenium**库

慢，很容易被反爬，且已经被许多网站反爬，网上各种对付反爬的解决办法基本不管用，不推荐

- 使用**pyppeteer**库

快，暂未被许多网站反爬，强烈推荐

03 | 用selenium库获取网页

selenium环境安装

```
pip install selenium
```

- 需要**chrome**浏览器或**firefox**浏览器，此外还需要下载**chrome**驱动程序（**chromedriver.exe**）或**firefox**驱动程序（**geckodriver.exe**）
- **安装方法：**

  https://googlechromelabs.github.io/chrome-for-testing/

  https://blog.csdn.net/m0_49449205/article/details/123971402?spm=1001.2014.3001.5501

用selenium获取网页

```
def getHtml(url):#暂时适用于百度图片搜索,慢且容易被反爬不是推荐的做法,随便看看
    from selenium import webdriver  #需要pip install selenium
    from selenium.webdriver.chrome.options import Options
    from selenium.webdriver.chrome.service import Service
    options = Options()  #浏览器选项
    #等价于 options = webdriver.chrome.options.Options()
    options.add_argument('—headless') #规定chrome浏览器隐身模式运行
    options.add_argument('--disable-gpu')
    #禁止chrome使用gpu加速,能快点
    service = Service(executable_path='./chromedriver.exe')
    driver = webdriver.Chrome( service=service, options=options )
    #driver就是个chrome浏览器,需要下载安装chrome驱动器chromedriver.exe
    driver.get(url)                          #浏览器装入网页
    html =  driver.page_source               #网页源代码
    driver.close()                           #关闭浏览器
    driver.quit()                            #退出
    return html                              #返回字符串
```


04 | 用pyppeteer库获取网页

pyppeteer 来历

- puppeteer是谷歌公司推出的可以控制Chrome浏览器的一套编程工具
- 一个日本工程师以此为基础推出了Python版本叫pyppeteer
- pyppeteer的官网在: https://pypi.org/project/pyppeteer/

pyppeteer工作原理

- 启动一个浏览器**Chromium**，用浏览器装入网页；浏览器可以用无头模式(**headless**)，即隐藏模式启动，也可以显式启动
- 从浏览器可以获取网页源代码，若网页有**javascript**程序，获取到的是**javascript**被浏览器执行后的网页源代码
- 可以向浏览器发送命令，模拟用户在浏览器上键盘输入，鼠标点击等操作，让浏览器转到其它网页
- **selenium**原理及功能和**pyppeteer**一样

pyppeteer环境安装

```
pip install pyppeteer
```

要求Python版本 >= 3.6

必须下载并安装特殊版本的谷歌浏览器Chromium

- 可以将Chromium压缩包随便解压在哪个文件夹, 然后在程序指明其中chrome.exe的位置,
- 也可以将Chromium解压到pyppeteer的安装文件夹下面, 这个文件夹通常类似:

  C:\Users\username\AppData\Local\pyppeteer\pyppeteer\local-chromium\588429
- 把username要换成自己的windows用户名, 588429这里可能是别的数;
- 将Chromium压缩包里面chrome-win32文件夹整个放在上面那个文件夹里面就行

强烈推荐作为爬虫首选工具!!!

预备知识：协程 (Coroutine)

协程是一种轻量级的线程，可以在单个线程内实现并发执行

与线程不同，协程的切换由程序控制（而非操作系统调度），避免了线程切换的开销

特点：

- 非抢占式：协程主动让出执行权 (通过 `await`)，而不是被强制中断
- 协作式调度：协程之间需要显式协作切换
- 适合I/O密集型任务：如网络请求、文件读写等

预备知识：协程的关键组件

(1) async 和 await 关键字

async：声明一个函数为异步函数（协程函数）

```
async def fetch_data():
    return "data"
```

await：挂起当前协程，等待另一个协程或异步操作完成

```
async def main():
    data = await fetch_data()  # 等待fetch_data()执行完毕
    print(data)
```

(2) 事件循环 (Event Loop)

协程的运行依赖于事件循环，负责调度协程的执行

预备知识: 协程的关键组件

(2) 事件循环 (Event Loop)

协程的运行依赖于事件循环, 负责调度协程的执行

```python
import asyncio

async def task():
    print("协程开始")
    await asyncio.sleep(1)  # 模拟I/O操作
    print("协程结束")

# 创建事件循环并运行协程
asyncio.run(task())  # Python 3.7+
```


预备知识: 协程

协程就是前面加了 **'async'** 的函数 (从Python 3.6开始有)

```
async def f()
        return 0
```

- **调用协程时, 必须在函数名前面加 'await'**

```
await f()
```

- 协程只能在协程里面调用, 即**await**语句只能出现在协程里面
- 协程是一种特殊的函数, 多个协程可以并行

pyppeteer中的所有函数都是协程, 调用时前面都要加 **await**, 且只能在协程中调用

- 初用协程, 经常因为调用XXX时忘了加 await 导致下面错误:

Runtime Warning: coroutine 'XXX' was never awaited

用pyppeteer获取网页

```
def getHtml(url): #暂时适用于百度图片搜索
        import asyncio                    #Python 3.6之后自带的协程库
        import pyppeteer a
        async def asGetHtm
            browser = awai
            # 启动Chromium,
            page = await b
            # 在浏览器中打开
            await page.set
            Win64; x64) Ap
            Chrome/78.0.390
            await page.evaluateOnNewDocument(
            '() =>{ Object.defineProperties(navigator, \
            { webdriver:{ get: () => false } }) }' )
            #反反爬措施
```

1. navigator.webdriver

- 是浏览器暴露的一个属性, 默认情况下, 当通过 Selenium/Puppeteer/Pyppeteer 等自动化工具控制浏览器时, 该属性会返回 true。
- 许多网站会检测 navigator.webdriver 来判断当前访问是否由自动化脚本发起, 如果是则可能拒绝服务或返回假数据。

2. Object.defineProperties

- 通过重新定义 navigator.webdriver 的 getter 方法, 强制其始终返回 false。
- 这样网站检测时会被欺骗, 认为这是普通用户的手动操作。

用pyppeteer获取网页

```
    await page.goto(url)   # 装入url对应的网页
    text = await page.content()
    # page.content就是网页源代码字符串
    await browser.close()   # 关闭浏览器
    return text
    #速度大约比用requests.get慢5,6倍

loop = asyncio.new_event_loop() # 创建新事件循环
asyncio.set_event_loop(loop)    # 设置为当前线程的事件循环
html = loop.run_until_complete(asGetHtml(url))
#返回值就是asGetHtml的返回值
return html
```

可以改进程序,只需要启动一次浏览器,生成一个page对象即可,以后获取不同网页都用相同page对象,所有事情完成后才关闭浏览器

用pyppeteer获取网页

launch的其它参数

```
browser = await launch(headless=False,
          executablePath = "c:/tmp/chrome-win32/chrome.exe",
          userdataDir = "c:/tmp")
```

- **excutablePath:** 如果Chromium没有安装在默认文件夹下面，则需要指定其位置
- **userdataDir:** userdataDir指明用来存放浏览器工作期间存放临时文件的文件夹;不是必须,能够防止可能出现的莫名其妙的错误

05 | 用BeautifulSoup库分析网页

分析并提取网页内容的三种方式

1. 正则表达式(速度最快,但适应变化略差)

2. **BeautifulSoup**库(速度是正则表达式的约几分之一)

3. **selenium**或**pyppeteer**的中的浏览器对象的查找元素函数(速度是正则表达式的约百分之一,用在需要模拟在网页中进行输入,点击按钮等操作的时候)

网页html文档中的tag

html文档中的tag

- tag格式通常为(少数没有正文和</X>):

```
<X attr1='xxx' attr2='yyy' attr3='zzz'...>
nnnnnnnnnnnnn
</X>
```

- X: tag的名字(name)
- attr1,attr2,…: tag的属性(attr)=后面跟着属性的值
- nnnnnnnnnnnnn: tag的正文(text)

html文档中的tag

- 例如：

```
<a href="www.sohu.com" id='mylink'>搜狐网</a>
```

- a：tag的名字(name)
- href, id：tag的属性(attr), =后面跟着属性的值
- 搜狐网：tag的正文(text)

tag可以嵌套

```
<div id="siteHeader" class="wrapper">
    <h1 class="logo">
    <div id="topsearch">
        <ul id="userMenu">
        <li ><a href="http://openjudge.cn/">首页</a></li>
    </div>
</div>
```


用BeautifulSoup库分析html

- 安装：

```
pip install beautifulsoup4
```

- 导入：

```
import bs4
```


用BeautifulSoup库分析html

- 使用：

1) 将**html**文档装入一个**BeautifulSoup**对象X

2) 用**X**对象的**find, find_all**等函数去找想要的**tag**对象

3) 对找到的**tag**对象，还可以用其**find, find_all**函数去找它内部包含（嵌套）的**tag**对象

4) **tag**对象的**text**就是该对象里的正文(**text**), **tag**对象也可以看作是一个字典, 里面包含各种属性(**attr**)及其值

把html文档载入BeautifulSoup对象

- 方法1 html文档来自字符串：

```
str = '''
<div id="siteHeader" class="wrapper">
        <h1 class="logo">
        <div id="topsearch">
                <ul id="userMenu">
                <li><a href="http://openjudge.cn/">首页</a></li>
        </div>
</div>
'''
# 带href的<a>都是链接,上面“首页”是链接文字, href后面
http://openjudge.cn是链接地址
soup = bs4.BeautifulSoup(str,"html.parser")
print(soup.find("li").text)      #>> 首页
```

把html文档载入BeautifulSoup对象

- 方法2 html文档来自于文件：

```
soup = bs4.BeautifulSoup(
       open("c:\\tmp\\test.html","r",encoding="utf-8"),
       "html.parser")
```


把html文档载入BeautifulSoup对象

- 方法3 html文档来自于给定网址：

```
import requests
def getHtml(url):
#获得html文本
        try:
                r = requests.get(url)
                r.raise_for_status()
                r.encoding = r.apparent_encoding
                return r.text
        except:
                return ""
```

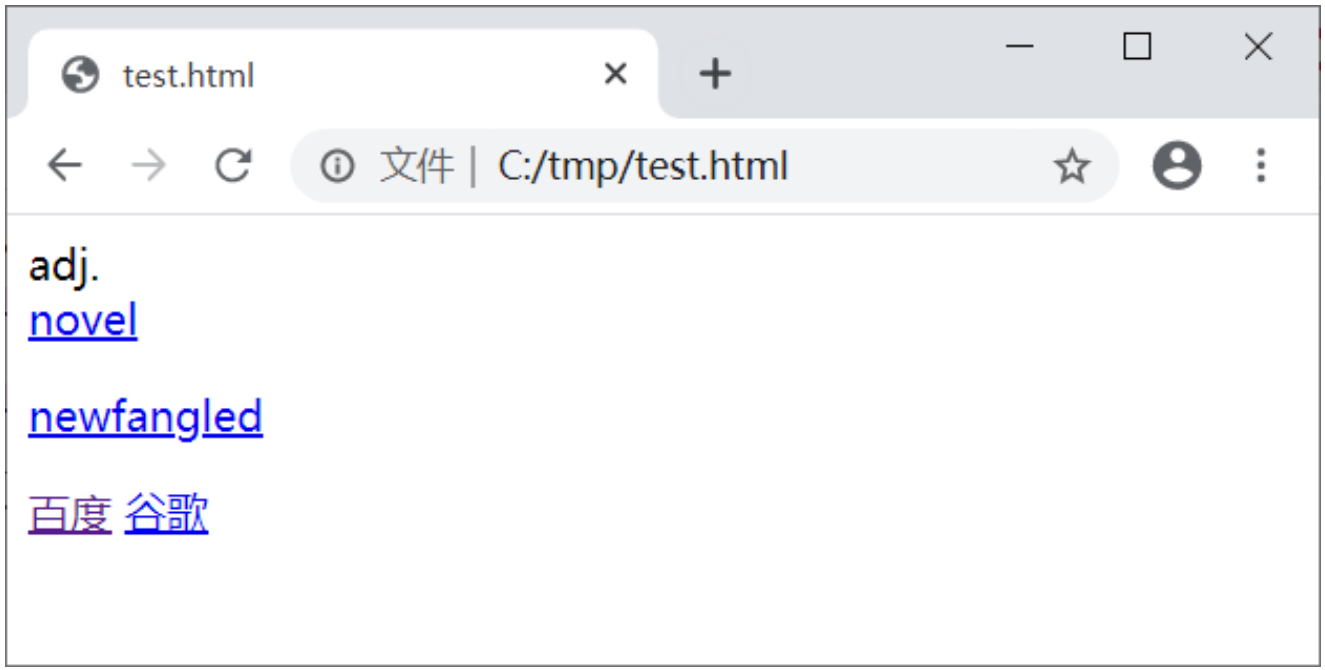
把html文档载入BeautifulSoup对象

- 方法3 html文档来自于给定网址：

```
html = getHtml("https://cn.bing.com/dict/search?q=new")

soup = bs4.BeautifulSoup(html, "html.parser")
```


c:\tmp\test.html:



用BeautifulSoup对象寻找想要的tag

c:\tmp\test.html:

```
<!DOCTYPE HTML>
<html>
<body>
<div id="synoid" style="display:block;">
        <div class="df_div2">
                <div class="de_title1">adj.</div>
                <div class="col_fl">
                        <a h="ID=Dictionary,5237.1"
                            href="https://cn.bing.com/dict/search?q=novel">
                             <span class="p1-4">novel</span>
                        </a>
                        <p>
```


用BeautifulSoup对象寻找想要的tag

c:\tmp\test.html:

```
                    <a h="ID=Dictionary,5238.1"
                    href="https://cn.bing.com/dict/search?q=newfangled">
                    <span class="p1-4">newfangled</span>
                    </a>
                </div>
        <a href="http://www.baidu.com" id="searchlink1" class="sh1">百度</a>
        <a href="http://www.google.com" id="searchlink1" class="sh2">谷歌</a>
        </div>
</div>
</body>
</html>
```


用BeautifulSoup对象寻找想要的tag

```
import bs4
soup = bs4.BeautifulSoup(open("c:\\tmp\\test.html",
                        encoding = "utf-8"),"html.parser")
diva = soup.find("div", attrs={"id":"synoid"})
#寻找名为"div",且具有 值为"synoid"的属性"id"的tag
if diva != None:  #如果找到
    for x in diva.find_all("span", attrs={"class":"p1-4"}):
        print(x.text)  #在diva内部继续找
    for x in diva.find_all("a", attrs={"id":"searchlink1"}):
        print(x.text)
    x = diva.find("a", attrs={"id":"searchlink1","class":"sh2"})
    if x != None:
        print(x.text)
        print(x["href"])
        print(x["id"])
```


用BeautifulSoup对象寻找想要的tag

输出：

```
novel
newfangled
百度
谷歌
谷歌
http://www.google.com
searchlink1
```


用BeautifulSoup对象寻找想要的tag

如果别处没有和
内部的模式相似的东西,也可以不用先找这个"synoid" tag

```
import bs4
soup = bs4.BeautifulSoup(
        open("c:\\tmp\\test.html",encoding="utf-8"), "html.parser")
for x in soup.find_all("span", attrs={"class":"p1-4"}):
    print(x.text)
for x in soup.find_all("a", attrs={"id":"searchlink1"}):
    print(x.text)
x = soup.find("a", attrs={"id":"searchlink1", "class":"sh2"})
if x != None:
    print(x.text)
    print(x["href"])
    print(x["id"])
```


06 | 实例:爬取每日股票交易信息

爬取每日股票交易信息


深振业A(000006)股票
上证所有股票代码及名
view-source:https://
创业板所有股票代码及
创业板所有股票代码及
/gupiao/list_sh.html
首页
概念股大全
个股资讯
A股要闻
龙虎榜
新股申购
新股中签查询
公司大全
近期热点
您的位置: 首页 » 股票大全 » 上证股票代码一览表
搜搜看 有没有你想找的
上证所有股票代码及名称
包钢股份(600010)
四川路桥(600039)
保利地产(600048)
九鼎投资(600053)
国投资本(600061)
皖维高新(600063)
凤凰光学(600071)
康欣新材(600076)
宋都股份(600077)
澄星股份(600078)
人福医药(600079)
东风科技(600081)
海泰发展(600082)
同济堂(600090)
恒力石化(600346)
明星电力(600101)
北方稀土(600111)
浙江东日(600113)
东方航空(600115)
浙江东方(600120)
商赢环球(600146)
航天机电(600151)
华升股份(600156)
巨化股份(600160)
天坛生物(600161)
香江控股(600162)
太原重工(600169)
美都能源(600175)
伊力特(600197)
金种子酒(600199)
江苏吴中(600200)
中金公司(601995)
福日电子(600203)
有研新材(600206)
安彩高科(600207)
新湖中宝(600208)
亚星客车(600213)
全柴动力(600218)
海航控股(600221)
鲁商发展(600223)

爬取每日股票交易信息

东方财富网每日股票交易信息单只股票：

quote.eastmoney.com/sh600000.html　上海证券

quote.eastmoney.com/sz000017.html　深圳或创业板

用pypetter爬取每日上证股票交易信息

单只股票： quote.eastmoney.com/sh600000.html

该页面查看源代码，看不到 12.17, 12.51等交易数据；说明源代码里面包含javascript程序，浏览器执行javascript程序以后，才能得到显示的页面；因此python程序需要在取到网页后，还要执行里面的javascript程序，才能得到股票数据

用requests.get无法得到显示的网页，必须用selenium或者pyppeteer

用 pyppeteer+正则表达式 爬取每日上证股票交易信息

```
html = getHtml("https://quote.eastmoney.com/sh600000.html")
# pypeteer版
# print(html)先打印出来,将打印结果拷贝粘贴到记事本,
# 查找关心的数据如12.17在哪里
```

```
class="quote3l_c"><div class="brief_info_c"><table><tbody><tr><td class="n">今开：</td><td><span><span
class="price_up blinkred">12.17</span></span></td><td class="n">最高：</td><td><span><span
class="price_up blinkred">12.51</span></span></td><td class="n">涨停：</td><td><span><span
class="price_up blinkred">13.34</span></span></td><td class="n">换手：</td><td><span><span
```

正则表达式匹配的格式：

```
<td class="n">XXXX</td><td><span><span class="XXXX">XXXX</span><
/span></td>
```


用 pyppeteer+正则表达式 爬取每日上证股票交易信息

```
html = getHtml("https://quote.eastmoney.com/sh600000.html")
# pypeteer版
# print(html)先打印出来,将打印结果拷贝粘贴到记事本,
# 查找关心的数据如12.17在哪里
pt = r'<td class="n">([^<]*)</td><td><span><span class[^<]*>
([^<]*)</span></span></td>'
for x in re.findall(pt, html, re.DOTALL):
    if (x[1] != ""):
        print(x[0], x[1])
```

re.DOTALL: 常规情况下.匹配任意字符但不包括换行符, 使用re.DOTALL后, .可以匹配换行符

正则表达式pt对应找到的关心的数据所在字符串的模式:

```
<td class="n">今开: </td><td><span><span class="price_up blink
red">12.17</span></span></td>
```


用 pyppeteer+正则表达式 爬取每日上证股票交易信息

输出结果：

```
今开：  6.61
最高：  6.63
涨停：  7.27
换手：  0.05%
成交量：  14.81万
总市值：  1934亿
昨收：  6.61
最低：  6.53
跌停：  5.95
量比：  1.86
成交额：  9724万
市净：  0.32
流通市值：  1934亿
```


用pyppeteer+BeautifulSoup爬取每日上证股票交易信息

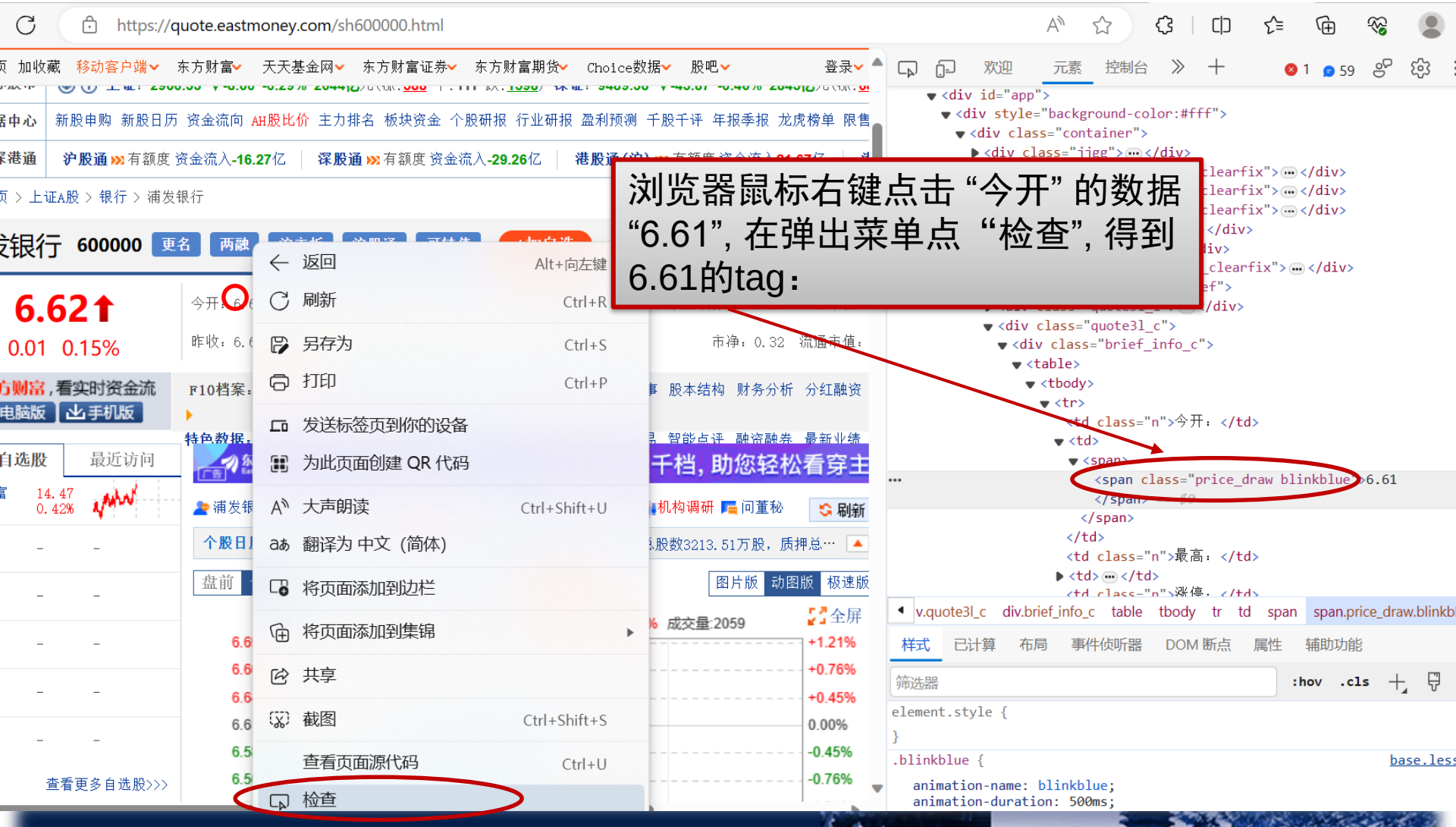
- 网页内容：

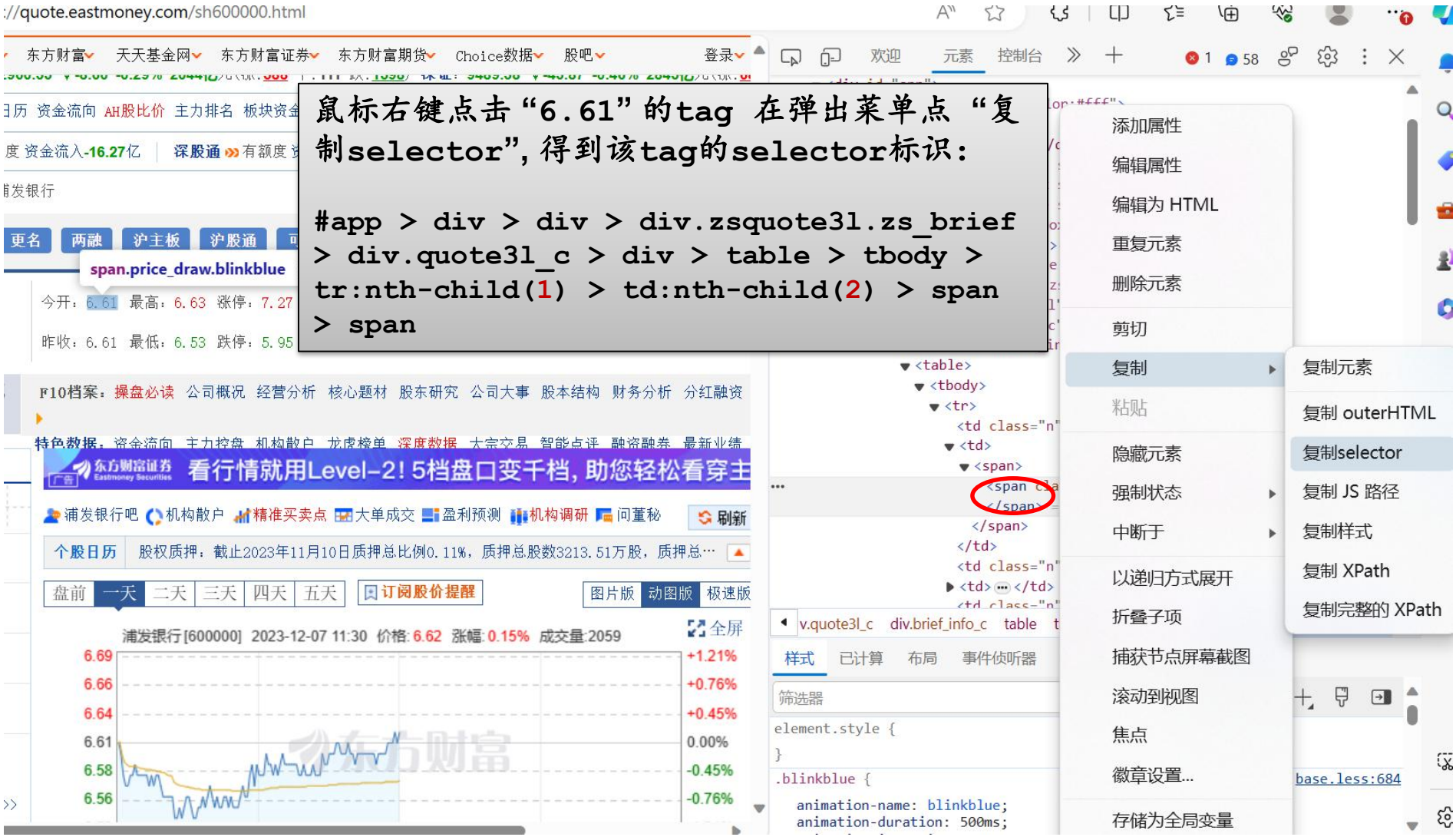
```
<div class="brief_info_c">
    <table>
        <tbody>
            <tr><td class="n">今开：</td><td><span><span class="price_up blinkred">11.73</span></span></td>
                <td class="n">最高：</td><td><span><span class="price_up blinkred">12.15</span></span></td>
                <td class="n">涨停：</td><td><span><span class="price_up blinkred">12.88</span></span></td>
                <td class="n">换手：</td><td><span><span class="price_draw blinkblue">0.29%</span></span></td>
                <td class="n">成交量：</td><td><span><span class="price_draw blinkblue">85.04万</span></span></td>
                <td class="n">市盈(动)
                    <span class="title_help">
                        <div class="ssy">
                            <ul>
                                <li class="ssy_t">·动态市盈率：<span class="price_draw">5.04</span></li>
                                <li>总市值除以全年预估净利润，例如当前一季度净利润1000万,则预估全年净利润4000万</li>
                                <li class="ssy_t">·静态市盈率：<span class="price_draw">7.83</span></li>
                                <li>总市值除以上一年度净利润</li>
                                <li class="ssy_t">·滚动市盈率：<span class="price_draw">7.80</span></li>
                                <li>最新价除以最近4个季度的每股收益</li>
                            </ul>
                        </div>
                    </span>：</td><td><span><span class="price_draw blinkblue">5.04</span></span></td>
                <td class="n">总市值：</td><td><span><span class="price_draw blinkblue">3546亿</span></span></td>
            </tr>
```


用pyppeteer+BeautifulSoup爬取每日上证股票交易信息

```
html = getHtml("https://quote.eastmoney.com/sh600000.html")
#要用pypeteer版的getHtml
#print(html) #下一步编程前可以先打印出来,
#将打印结果拷贝粘贴到记事本,查找关心的数据如12.17在哪里
soup = bs4.BeautifulSoup(html,"html.parser")
diva = soup.find("div", attrs={"class":"brief_info_c"})
titles = diva.find_all("td", attrs = {"class":"n"})
for t in titles:
    sp = t.findNext("span")#从当前标题<td>向后查找第一个<span>
    if 'class' not in sp.attrs:
        print(t.text, sp.text)
    #print(sp)可以看到sp的整个tag的描述,如:
    <span><span class="price_up blinkred">0.01</span>
```

用pyppeteer+浏览器查找元素功能爬取每日上证股票交易信息




鼠标右键点击“6.61”的tag 在弹出菜单点 “复制selector”,得到该tag的selector标识:
#app > div > div > div.zsquote3l.zs_brief > div.quote3l_c > div > table > tbody > tr:nth-child(1) > td:nth-child(2) > span > span
span.price_draw.blinkblue
今开：6.61 最高：6.63 涨停：7.27
昨收：6.61 最低：6.53 跌停：5.95
添加属性
编辑属性
编辑为 HTML
重复元素
删除元素
剪切
复制
粘贴
隐藏元素
强制状态
中断于
以递归方式展开
折叠子项
捕获节点屏幕截图
滚动到视图
焦点
徽章设置...
存储为全局变量
复制元素
复制 outerHTML
复制selector
复制 JS 路径
复制样式
复制 XPath
复制完整的 XPath

通过pypeteer的Page对象的querySelector函数可以由selector(字符串)得到对应的元素

今开 的selector:

```
#app > div > div > div.zsquote3l.zs_brief > div.quote3l_c > div > table > tbody > tr:nth-child(1) > td:nth-child(1)
```

6.61 的selector::

```
#app > div > div > div.zsquote3l.zs_brief > div.quote3l_c > div > table > tbody > tr:nth-child(1) > td:nth-child(2) > span > span
```

最高 的selector:

```
#app > div > div > div.zsquote3l.zs_brief > div.quote3l_c > div > table > tbody > tr:nth-child(1) > td:nth-child(3)
```

6.63 的selector:

```
#app > div > div > div.zsquote3l.zs_brief > div.quote3l_c > div > table > tbody > tr:nth-child(1) > td:nth-child(4) > span > span
```

通过pypeteer的Page对象的querySelector函数可以由selector(字符串)得到对应的元素

昨收 的selector:

```
#app > div > div > div.zsquote3l.zs_brief > div.quote3l_c > div > table > tbody > tr:nth-child(2) > td:nth-child(1)
```

第二行的6.61 的selector::

```
#app > div > div > div.zsquote3l.zs_brief > div.quote3l_c > div > table > tbody > tr:nth-child(2) > td:nth-child(2) > span > span
```

用pyppeteer+浏览器查找元素功能爬取每日上证股票交易信息

```
import asyncio  #Python 3.6之后自带的协程库
import pyppeteer as pyp
import re
async def antiAntiCrawler(page):    #为page添加反反爬虫手段
    await page.setUserAgent('Mozilla/5.0 (Windows NT 6.1; Win64; x64) \
                        AppleWebKit/537.36 (KHTML, like Gecko) \
                        Chrome/78.0.3904.70 Safari/537.36')
    await page.evaluateOnNewDocument(
        '() =>{ Object.defineProperties(navigator, \
        { webdriver:{ get: () => false } }) }')
```


用pyppeteer+浏览器查找元素功能爬取每日上证股票交易信息

```
async def asGetStockInfo(url):
    # url是"https://quote.eastmoney.com/sh600000.html"
    browser = await pyp.launch(headless=False)
    page = await browser.newPage()
    await antiAntiCrawler(page)
    await page.goto(url)
    html = await page.content()
    for k in range(1,3):
        for i in range(1,14,2):
            selector1 = '#app > div > div > div.zsquote3l.zs_brief ' + \
            '> div.quote3l_c > div > table ' + \
            '> tbody > tr:nth-child(%d) > td:nth-child(%d)' % (k, i)
            selector2 = ('#app > div > div > div.zsquote3l.zs_brief' + \
             '> div.quote3l_c > div > table > tbody > tr:nth-child(%d)' + \
            '> td:nth-child(%d) > span > span') % (k, i+1)
```

用pyppeteer+浏览器查找元素功能爬取每日上证股票交易信息

```
            title = await page.querySelector(selector1)
            if title is None:
                break
            else:
                value = await page.querySelector(selector2)
            obj = await title.getProperty("innerText")
            text = await obj.jsonValue()  # 固定写法
            print(text, end = " ")
            obj = await value.getProperty("innerText")
            text = await obj.jsonValue()  # 固定写法
            print(text)
    await browser.close()

#以下是调用协程的固定写法
loop = asyncio.new_event_loop()
asyncio.set_event_loop(loop)
url = "https://quote.eastmoney.com/sh600000.html"
loop.run_until_complete(asGetStockInfo(url))
```


07 | 需要登录的爬虫

需要登录的爬虫

- 许多网站需要登录后才能访问其内容

  京东、淘宝需要登录才能访问交易记录

  openjudge.cn 需要登录才能看提交过的源代码

- 登录操作，无法用一个url表示出来
- 解决办法之一：用浏览器模拟登录过程，输入用户名密码、点登录按钮，或者程序启动浏览器，等待手工登录后，程序再继续爬虫操作 (对有验证码的情况)

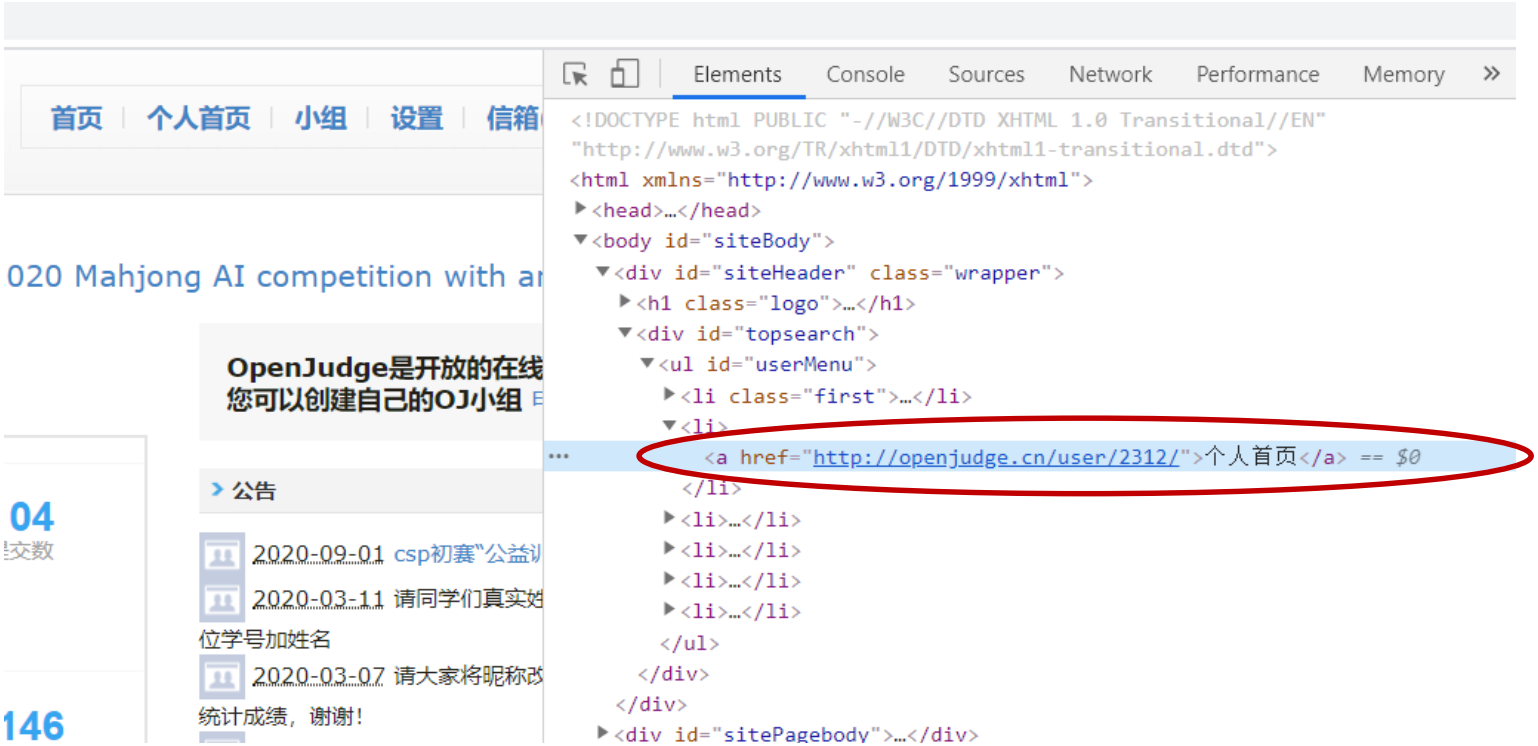
爬取Openjudge自己提交通过的所有程序源码

- 程序命令浏览器模拟登录过程，即输入用户名密码，点登录按钮
- 或 程序启动浏览器，等待手工登录后，程序再继续爬虫操作 (对有验证码的情况，或者懒得写代码的情况)

- 更高级做法：不用浏览器，经数据包分析后，用requests库进行数据传输进行登录

爬取Openjudge自己提交通过的所有程序源码

鼠标右键点击右上角的“个人首页”，在弹出的菜单上选“检查”：



爬取Openjudge自己提交通过的所有程序源码

点击“个人首页”，进入：


```
<a href="http://cxsjsxmooc.openjudge.cn/2020t1fallall2/solution/25212869/"
    class="result-right">Accepted</a>
```

爬取Openjudge自己提交通过的所有程序源码

点击某个题的 Accepted 链接, 进入:


```
<pre class="sh_python">
n = int(input())
lst = []
for i in range(n):
    s = input().split()
    lst.append((s[0], int(s[1])))
lst.sort(key= lambda x:(-x[1], x[0]))
for x in lst:
    print(x[0], x[1])</pre>
```

pyppeteer爬取Openjudge自己提交通过的所有程序源码

```
import asyncio
import pyppeteer as pyp
async def antiAntiCrawler(page):
    #为page添加反反爬虫手段
    await page.setUserAgent('Mozilla/5.0 (Windows NT 6.1; Win64; x64) \
                            AppleWebKit/537.36 (KHTML, like Gecko) \
                            Chrome/78.0.3904.70 Safari/537.36')
    await page.evaluateOnNewDocument(
        '() =>{ Object.defineProperties(navigator, \
        { webdriver:{ get: () => false } }) }')
```

pyppeteer爬取Openjudge自己提交通过的所有程序源码

```
async def getOjSourceCode(loginUrl):
    width, height = 1400, 800  #网页宽高
    browser = await pyp.launch(headless = False,
                               userdataDir = "c:/tmp",
                               args = [f'--window-size={width},{height}'])
    page = await browser.newPage()
    await antiAntiCrawler(page)
    await page.setViewport({'width': width, 'height': height})
    await page.goto(loginUrl)
   #若手动登录,则以下若干行可以去掉
    element = await page.querySelector("#email")      #找到账户输入框
    await element.type("XXXXX@pku.edu.cn")            # 输入邮箱
    element = await page.querySelector("#password")   #找到密码输入框
    await element.type("XXXXXXXXX")                   # 输入密码
```

pyppeteer爬取Openjudge自己提交通过的所有程序源码

```
    element = await page.querySelector("#main > form > div.user-login > p:nth-child(2) > button")  #找到登录按钮
    await element.click()  #点击登录按钮
    #若手动登录,则以上若干行可以去掉
    await page.waitForSelector("#main>h2", timeout=30000)
    #等待“正在进行的比赛...”标题出现
    element = await page.querySelector("#userMenu>li:nth-child(2)>a")
    #找“个人首页”链接
    await element.click()                    #点击个人首页链接
    await page.waitForNavigation()  #等新网页装入完毕
    elements = await page.querySelectorAll(".result-right")
    #找所有 Accepted 链接,其有属性 class="result-right"
    page2 = await browser.newPage() #新开一个页面(标签)
    await antiAntiCrawler(page2)
```


pyppeteer爬取Openjudge自己提交通过的所有程序源码

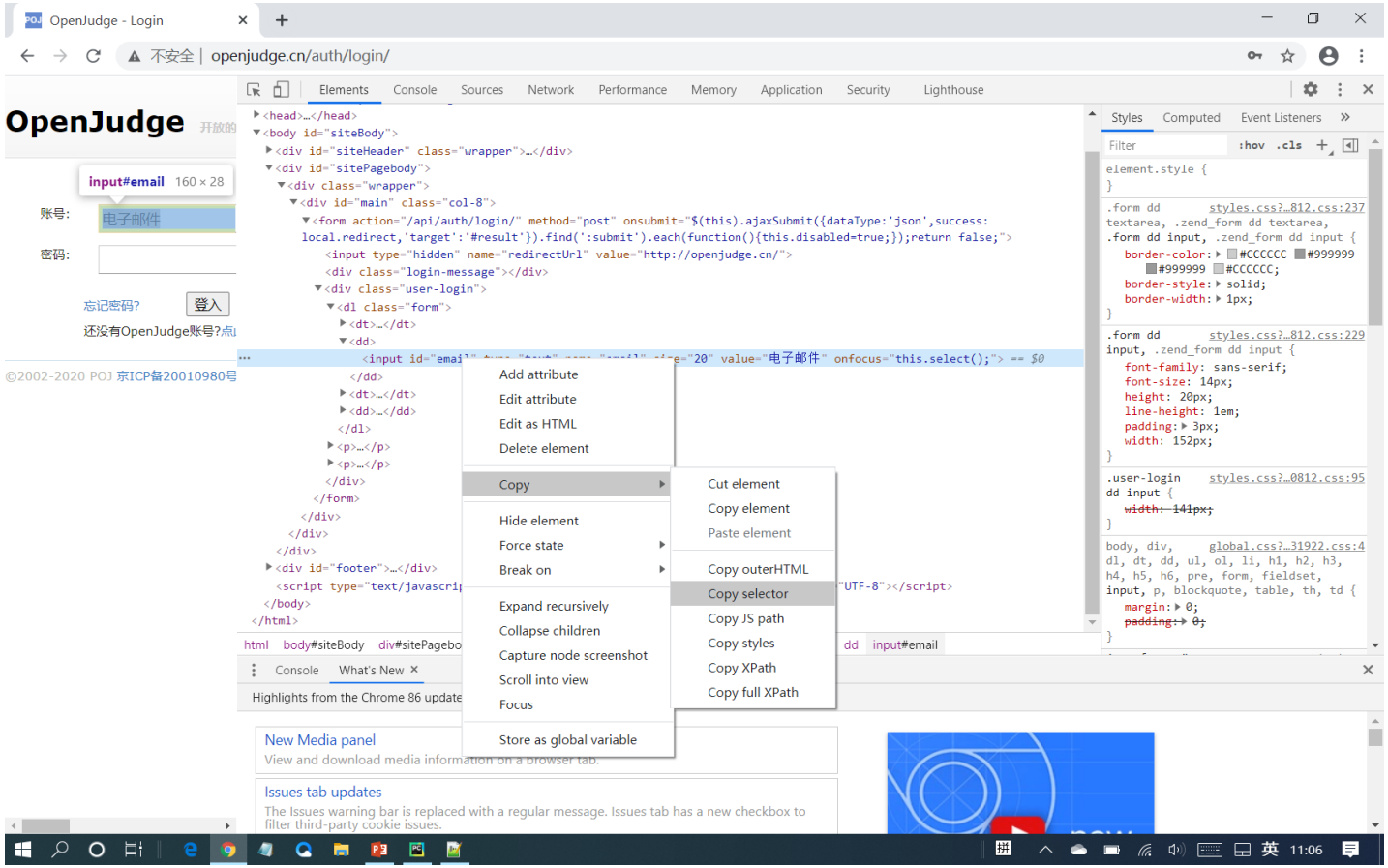
```
    for element in elements[:2]: #只打印前两个程序
        obj = await element.getProperty("href") #获取href属性
        url = await obj.jsonValue()
        await page2.goto(url)       #在新页面(标签)中装入新网页
        element = await page2.querySelector("pre") #查找pre tag
        obj = await element.getProperty("innerText") #取源代码
        text = await obj.jsonValue()
        print(text)
        print("-------------------------")
    await browser.close()
def main():
    url = "http://openjudge.cn/auth/login/"
    asyncio.get_event_loop().run_until_complete(getOjSourceCode(url))
main()
```


pyppeteer爬取Openjudge自己提交通过的所有程序源码

获取tag的selector

得到selector：

#email



08 | 补充tips

绝对网址和相对网址

绝对网址以 http:// 或 https:// 开头，相对网址无这两种开头

如果当前网页网址是：

http://www.pku.edu.cn/education/index.htm

而该网页中有一个链接，其中网址是相对的，形如：

```
<a href="dict/word.htm">词典单词</a>
```

则该链接的真实网址（绝对网址）是：

http://www.pku.edu.cn/education/dict/word.htm

绝对网址和相对网址

使用requests库时，获得当前网页网址：

```
r = requests.get("http://openjudge.cn")
print(r.url) #>>http://openjudge.cn
```

或：

```
session = requests.session()
r = session.get("http://openjudge.cn")
print(r.url)
```

使用pyppeteer库时，获得当前网页网址：

```
browser = await pyp.launch(headless=False)
page = await browser.newPage()
await page.goto("http://openjudge.cn")
print(page.url)              #>>http://openjudge.cn
```


反反爬技巧

连续的两个操作之间，加入适当延时，模拟人的动作，以免因动作太快被识破

```
import time
time.sleep(2)    #暂停2秒，啥也不做
```

也可以用time.sleep(...)来等待一段时间，确保网页加载完成