XPath常用规则：

nodename: 选取此节点的所有子节点

/ : 从当前结点选取直接子节点

// : 从当前节点选取子孙节点(可以不是相邻的节点)

. : 选取当前节点

.. : 选取当前节点的父节点

@ : 选取属性 //li/a/@href 选取所有节点下的a节点中的href属性值

//\* : 选取所有节点

选出文本： //li[@class=”item-0”]/a/text() 选出所有li,且class属性值为item-0的节点下的a节点下的文本值

选出属性：@

value1 = html.xpath(‘//a/@href’) 选出节点a中的href属性

value2 = html.xpath(‘//img/@src’) 选出节点img中的src属性

value3 = html.xpath(‘//div[2]/span/@id’) 选出div[2]节点下的span节点中的id属性

一些节点的属性有多个值： <li class=” li li-first li-second”>。需要用contains()匹配。

//li[contains(@class, “li”)] /a/text() contains(属性名称， 属性值)

多属性匹配：根据多个属性确定一个节点(需要匹配多个属性)<li class=”li li-first” name=”item”><a….>

xpath(‘//li[contains(@class, ”li”) and @name= “item”]/a/text()’)同时匹配class和name属性

按序选择：在选择的时候，同时匹配了多个节点，但只想要其中的某个节点(如第2个，或最后一个节点)，可以利用中括号传入索引的方法获取特定次序的节点。

result = html.xpath(‘//li[last()]/a/text()’) result = html.xpath(‘//li[last()-2]/a/text()’)

Beautiful soup:python的一个HTML或XML的解析库：可以方便地从网页中提取数据。依赖于lxml库

find\_all() : 查询所有符合条件的元素组成的列表。find\_all(name节点名, attrs, recursive, text, \*\*kwargs)

find(): 查询返回单个元素(第一个匹配的元素)

find\_parents() , find\_parent() , find\_next\_siblings() , find\_next\_sibling() , find\_previous\_siblings() , find\_previous\_sibling()

动态页面抓取：

Ajax: 异步的javascript和XML：利用js在保证页面不被刷新，页面链接不改变的情况下与服务器交换数据并更新部分网页的技术(传统网页，如果要更新其内容，那么必须刷新整个页面)。

CSS选择器(id和class选择器)：如果要在html元素中设置CSS样式，需要在元素中设置id和class选择器。Id选择器为特定id的html元素指定特定的样式。Class选择器在多个元素中使用，也可以指定特定的html元素使用。

Selenium: 自动化测试工具，利用它可以驱动浏览器执行特定的动作，同时还可以获取浏览器当前呈现的页面的源代码。对于js动态渲染的页面而言，此种抓取方式非常有效。

查找节点：单个节点

Selenium提供了8种定位方式：

find\_element\_by\_id

find\_element\_by\_name

find\_element\_by\_xpath driver.find\_element\_by\_xpath(“//input[@id=’kw’ and @name=’wd’]”)

find\_element\_by\_link\_text 通过完整的超链接定位

find\_element\_by\_partial\_link\_text 通过部分超链接定位

find\_element\_by\_tag\_name

find\_element\_by\_class\_name

find\_elements\_by\_css\_selector

延时等待：隐式等待；显示等待

隐式等待：当使用隐式等待时，如果selenium没有在DOM中找到节点，将继续等待，超出时间后，则抛出找不到的异常。browser.implicitly\_wait(10)

显示等待(更加合适的等待方法)：指定要查找的节点，然后指定一个最长等待时间。

Scrapy框架：纯python实现的爬虫框架，模块之间耦合程度底，可扩展性强。

Engine:

Item: 保存爬取数据的容器(定义爬取数据的数据结构)

Scheduler:

Downloader:

Spiders: 定义爬取网站的动作。分析爬取下来的网页。以初始的URL初始化request,并设置回调函数。当request成功请求并返回时，response生成并作为参数传给该回调函数。

Item pipeline: 当spider解析完response之后，item就会传递到item pipeline.item pipeline组件会依次调用，完成数据清洗，存储等(清理html数据，验证爬取数据，检查爬取字段，查重，丢弃重复内容，保存数据库)。

ImagesPipeline是scrapy自带的类，用来处理图片(爬取时将图片下载到本地)

Downloader middlewares: scheduler从队列中拿出一个request发送给Downloader执行下载前，会先经过Downloader Middleware的处理。另外当Downloader将Request下载完成得到的Response返回给Spider前会再次经过Middleware的处理。主要负责：修改User-Agent,处理重定向，设置代理，失败重试，设置cookies

process\_request(request,spider): request被scrap引擎调度给downloader之前，process\_request()方法会被调用(在下载前会被调用)，此时process\_request()方法可以对request进行处理。

Spider middlewares: