1. **正则表达式：**

. ：匹配任意一个字符（可用于**匹配中文字符**）, eg（两点）: ..匹配任意数量的字符

\w：匹配任意一个数字、字母（非特殊字符）

\d：匹配十进制数

\s：空白字符

\W：匹配\w相反的字符

\S： 匹配\s相反的字符

^：匹配开始位置

$：匹配结束位置

t{n}：前面的原子出现n次 // 若为t{n,}：至少出现n次

|：(也就是**或**)模式选择符：t|s，选择t或者s

( )：只输出（）内的内容，其余的括号外的表达式用于定位

[ ]：

\*?：重复任意次，但尽可能少重复   
+?：重复1次或更多次，但尽可能少重复   
??：重复0次或1次，但尽可能少重复   
{n,m}?：重复n到m次，但尽可能少重复   
{n,}?：重复n次以上，但尽可能少重复

**.\*? ：匹配任意字符**

**(.\*?)：只输出括号内的数据**

**eg：**

**#使用正则表达式**

**import re**

pat3 = "python\*"

string3 = "ijsadpythonasdf"

res3 = re.search(pat3,string3)

print(res3)

<re.Match object; span=(5, 11), match='python'>

1. **模式修正符：**在不修改表达式的情况下，改变情况

I：忽略大小写

M

L

S

**eg：**

pat4 = "python"

string4 = "Python-asffdd"

res4 = re.search(pat4, string4, **re.I**)

print(res4)

<re.Match object; span=(0, 6), match='Python'>

1. **贪婪模式、懒惰模式：**
2. 贪婪模式：尽可能地匹配多的字符：覆盖广，.\*
3. 懒惰模式：只匹配少的字符：精准定位，.\*?
4. **常用的表达式：** **(.\*)**、**(.\*?)**、**.\***、**.\*?**

eg：

#贪婪模式

pat1 = "**p.\*y**"

#懒惰模式

pat2 = "**p.\*?y"**

string = "py-asd-python-py-"

res1 = re.search(pat1, string)

res2 = re.search(pat2, string)

print(res1)

print(res2)

<re.Match object; span=(0, 16), match='**py-asd-python-py**'>

<re.Match object; span=(0, 2), match='**py**'>

1. **sub()、match()、search():**

sub()：用于替换  
match()：重头开始匹配

search()：

1. **匹配网站数据的例子：**

pat = "[a-zA-Z]+://[^\s]\*[.com|.cn]"

string = '<a href=http://www.baidu.com>asdasf<\a>'

res = **re.compile(pat).findall(string) #规则+ 数据**

print(res)

['http://www.baidu.com']

1. **爬取网站的QQ号：**

import re

import urllib.request

# pat = "<p>**\d(.\*?)\d**</p>" #只会显示2184083，因为（）的存在

**#懒惰模式**

pat = "<p>**(\d.\*?\d)**</p>"

**#爬取网址的数据**

data = **urllib.request.urlopen**("https://edu.csdn.net/huiyiCourse/detail/215")**.read()**

res = **re.compile(pat).findall(str(data))** #此处必须使用str类型

print(res)

['521840835']

1. **urllib库使用：爬取网页数据使用**
2. urlretrieve()：将网页保存到本地
3. urlcleanup()：清除urlretrieve产生的缓存
4. geturl()：获得网址
5. infon()：获得网页信息
6. getcode()：获得网页状态码

**eg:**

import urllib.request

data = **urllib.request.urlretrieve**("http://read.douban.com/provider/all", **filename =** "./result/douban.html")

**#爬取网页数据：timeout,反应时间**=1，超时就会报错

data\_1 = **urllib.request.urlopen**("http://read.douban.com/provider/all",**timeout =** 1)

**#清除缓存**

urllib.request.urlcleanup()

website = data\_1.**geturl**()

print(website) #<https://read.douban.com/provider/all>

websiteInfo = data\_1.**info**()

1. **quote()：解决编码问题**，当搜索中出现中文时使用

chinese = urllib.request.quote("习近平")

url\_chg = "http://www.google.com/search?q=" + chinese

req = urllib.request.**Request**(url\_chg)

data\_chinese = urllib.request.urlopen(req).read()

eg：

keyname = "鞋子"

key = urllib.request.quote(keyname)

print(key)

%E9%9E%8B%E5%AD%90

1. **自动模拟HTTP请求：**将爬虫伪装成浏览器，防止被网站屏蔽程序
2. **re.S：匹配多行**

eg：data = re.compile(url, re.S).read()

1. 异常处理：

URLError：无异常状态码（包含HTTPError）

HTTPError：有异常状态码：

产生异常的情况：

1. 连不上服务器
2. URL不存在
3. **爬取新闻例程：**

import urllib.request

import re

**#1、先获取新闻首页的数据**

data = urllib.request.urlopen("https://news.163.com/").read()

**#2、编码**

data\_decode = data.decode("utf-8","ignore")

**#3、将新闻首页获得的内容，使用正则表达式找出各个新闻**

**#这里的https://有时候会出错，可以使用http://**

pat = 'href="([https://news.163.com/.\*?)](https://news.163.com/.*?))"'

all\_url = re.compile(pat).findall(data\_decode)

**#4、逐条保存新闻**

for i in range(0, len(all\_url)):

try:

print("th:" + str(i) + " time" )

thisurl = all\_url[i]

filename = "./result/163news/" + str(i) + ".html"

**#保存数据：网址 + 保存的路径（文件名）**

**urllib.request.urlretrieve**(thisurl, **filename =** filename)

print("success")

except urllib.error.URLError as e:

if hasattr(e, "code"):

print(e.code)

if hasattr(e, "reason"):

print(e.reason)

2. **爬虫：代理服务器**

爬虫防止浏览器屏蔽手段：

1. 伪装成浏览器
2. 使用代理服务器：

使用不同服务器的不同IP来解决网页禁止同一IP的多次访问的问题。

**eg:**

**#代理服务器示例代码**

import urllib.request

def user\_proxy(url, IP):

**#代理服务器**

**proxy = urllib.request.ProxyHandler({"http:":IP})**

opener = urllib.request.build\_opener(proxy, urllib.request.HTTPHandler)

#安装为全局变量

urllib.request.install\_opener(opener)

#爬取网页数据

data = urllib.request.urlopen(url).read().decode("utf-8","ignoer")

return data

proxy\_addr = "1.197.203.22:9999"

url = "https://map.baidu.com/@11590057.96,4489812.75,4z"

data = user\_proxy(url, proxy\_addr)

1. **图片爬取：**

**想找到高清的图片，在网页源码中查找（pic\_url":"//）**

import urllib.request

import re

keyname = "鞋子"

key = urllib.request.quote(keyname)

print(key)

#伪装成浏览器

headers = ("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36")

opener = urllib.request.build\_opener()

opener.addheaders = [headers]

urllib.request.install\_opener(opener)

#爬取每页的图片地址

for i in range(0,3):

url="https://s.taobao.com/search?q="+key +"&imgfile=&js=1&stats\_click=search\_radio\_all%3A1&initiative\_id=staobaoz\_20190825&ie=utf8&bcoffset=3&ntoffset=3&p4ppushleft=1%2C48&s="+str(i\*44)

data = urllib.request.urlopen(url).read().decode("utf-8","ignore")

pat = 'pic\_url":"//(.\*?)"'

imageList = re.compile(pat).findall(data)

print(len(imageList))

for j in range(0, len(imageList)):

thisImage = imageList[j]

#得到每张图片的地址，使用http协议打开

thisImageUrl = "http://" + thisImage

fileName = "./result/img/" + str(i) + str(j) + ".jpg"

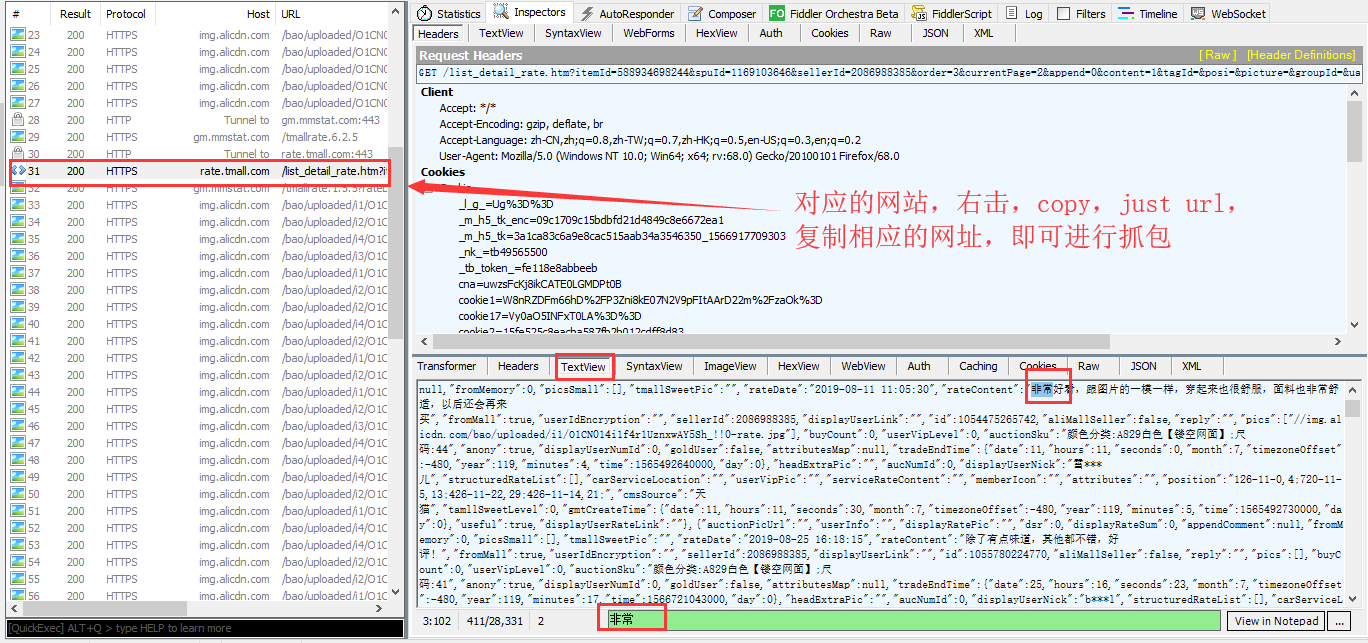
urllib.request.urlretrieve(thisImageUrl, filename = fileName)

1. **抓包：（其实不好用，启用fidder，会使jupyter无法urlopen获取网页数据，必须关闭fidder后再运行程序）**
2. 先配置fidder：<https://www.cnblogs.com/woaixuexi9999/p/9247705.html>

配置浏览器代理时，需要cmd查看IP：ipconfig，再进行IP、端口设置

（在启动fidder时，会提示是否随机port，选择否，就可以自己在tools中配置port）

**本质上：**browser ——>fidder代理——>联网

1. fidder具体操作：
2. **多线程爬虫：**

**采用多线程的方式来运行多个爬虫代码**

**import** threading  
  
**class Thread\_1**(threading.Thread):  
 **def** \_\_init\_\_(self):  
 threading.Thread.\_\_init\_\_(self)  
 **def run**(self):  
 **for** i **in** range(0, 10):  
 print("Thread 1", i)  
  
**class Thread\_2**(threading.Thread):  
 **def** \_\_init\_\_(self):  
 threading.Thread.\_\_init\_\_(self)  
 **def run**(self):  
 **for** i **in** range(0, 10):  
 print("Thread 2:", i)  
  
thread1 = Thread\_1()  
thread2 = Thread\_2()  
thread1.start()  
thread2.start()

在pycharm中运行，无法得到现象，使用Jupyter notebook可以得到现象

1. **Scrapy框架**
2. **通用命令：**

scrapy fetch + URL（网站）：查看爬取的过程

scrapy run：运行一个爬虫文件

scrapy shell + URL：进入交互终端

**scrapy startproject**：创建爬虫项目**（常用）**

scrapy view + URL：打开、下载网站

scrapy genspider –l：显示当前可以使用的爬虫项目模板

scrapy genspider –t basic weixin baidu.com：基于basic模板创建项目weixin，只

爬取百度的数据。

scrapy check：测试爬虫代码

**scrapy crawl weixin**：运行、启动爬虫文件**（常用）**

scrapy list：当前项目可用的爬虫文件

scarpy parse + URL：获取某个网页的数据

**--nolog：**在命令尾部加上这个，就不会显示提示信息

1. **Scrapy工程创建参考：**

<https://blog.csdn.net/water3821/article/details/79970621>

使用scrapy startproject时，

出现错误：scrapy startproject xxx 报错, ImportError: DLL load failed:

解决方案：在cmd下：pip install -I cryptography

**创建工程时**，必须先用可在pycharm上的cmd 命令：scrapy startproject 工程名

**运行工程：**可用scrapy crawl 文件名 （也可以自己写一个.py文件来运行）

eg：**写一个main.py作为工程运行的主函数**

**from** scrapy.cmdline **import** execute  
**import** os  
**import** sys  
  
sys.path.append(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
#显示执行的提示详情  
#execute(["scrapy", "crawl", "TEST\_FUNCTION"])  
#只显示结果  
execute(["scrapy", "crawl", "TEST\_FUNCTION", "--nolog"])

1. **Xpath表达式：**效率高于正则表达式

（1）/html/head/title：提取该目录下的内容。

<html class=“js”>

<head>

……

（2）/：从顶端往下依次寻找

text()：提取文本信息

@：定位属性，class、href…

//：寻找标签：eg：<li、<a…

1. **scrapy工程文件说明：**

**使用工程的步骤如下：**

1、items.py：定义数据存放的格式

2、middlewares.py：

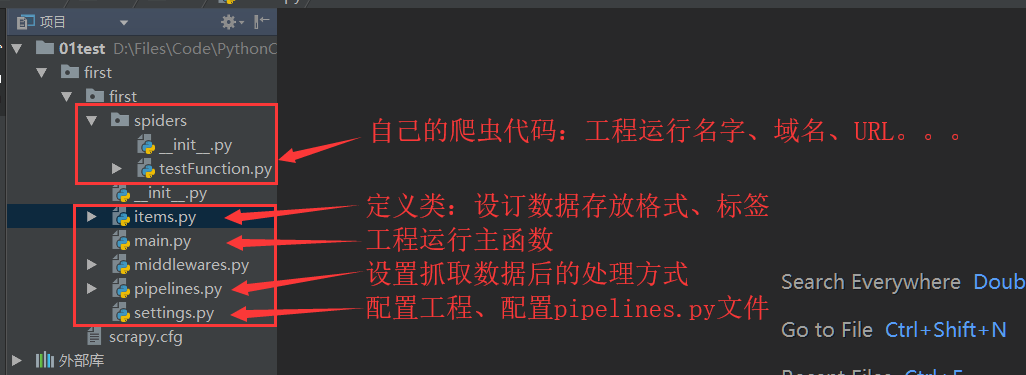
3、pipelines.py：设置爬取数据之后的处理方式

4、setting.py：配置pipelines.py

eg：在first工程中，pipelines.py内的FirstPipeline，300是一个序号

ITEM\_PIPELINES = {  
 'first.pipelines.FirstPipeline': 300,  
}

5、spiders文件夹：存放自己的代码，编写爬虫代码



1. **scrapy所有输出的结构都存在response对象中**。

xpath使用：response.xpath().extract()

**eg：**新浪新闻网页源码：

<html>

<head>

<meta http-equiv="Content-type" content="text/html; charset=utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<title>新浪首页</title>

想获取：新浪首页 ， 文字

xpath实现形式：/html/head/title/text()

**scrapy完整实现形式：response用于存放获得的数据**

response.xpath("/html/head/title/text()").extract()

1. **伪装成浏览器 + 使用代理服务器**

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
'''  
使用代理服务器 + 伪装成浏览器  
'''  
\_\_author\_\_ = 'Chris'  
**import** urllib.request  
**import** urllib.error  
**import** time  
**def use\_proxy**(url, IP):  
 **try**:  
 #伪装成浏览器  
 req = urllib.request.Request(url)  
 req.add\_header('User-Agent', 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36')  
 #代理服务器  
 proxy = urllib.request.ProxyHandler({'http:':IP})  
 opener = urllib.request.build\_opener(proxy, urllib.request.HTTPHandler)  
 #安装为全局变量  
 urllib.request.install\_opener(opener)  
 #爬取网页数据  
 data = urllib.request.urlopen(req).read().decode("utf-8","ignore")  
 **return** data  
 **except** urllib.error.URLError **as** e:  
 **if** hasattr(e, "code"):  
 print("E.code:" + e.code)  
 **if** hasattr(e, "reason"):  
 print("E.REASON" + e.reason)  
 #若为URLError异常，延时10s执行  
 time.sleep(10)  
 **except** Exception **as** e:  
 print("EXCEPTION :" + str(e))  
 #若为Exception异常，延时1s执行  
 time.sleep(2)  
  
**if** \_\_name\_\_ == "\_\_main\_\_":  
 # proxy\_addr = "1.197.203.22:9999"  
 proxy\_addr = "110.86.136.89:9999"  
 url = "https://map.baidu.com/@11590057.96,4489812.75,4z"  
 data = use\_proxy(url, proxy\_addr)  
 print(data)  
 print(len(data))