1 requests库

1.1 get()

- r = requests.get(url,params=None,**kwargs)
- r:访问url返回的内容
- params:url中的额外参数，字典或字节流格式，可选
- **kwargs： 12个控制访问的参数

属性	说明
r.status_code	HTTP请求的返回状态
r.text	HTTP相应内容的字符串形式，即url对应的页面内容
r.encoding	从HTTP header中猜测的相应内容编码方式
r.apparent_encoding	从内容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

- r.encoding：如果header中不存在charset,则认为编码为ISO-8859-1
- r.apparent_encoding：根据网页内容分析出的编码方式

In [ ]:

```
import requests
r=requests.get('https://www.52pojie.cn')
print(r.status_code)
print(r.encoding)
print(r.apparent_encoding)
print(r.text)
```

1.1.1 r.text 与 r.content 的区别

- 简单粗暴来讲：
  - text 返回的是unicode 型的数据，一般是在网页的header中定义的编码形式。
  - content返回的是bytes，二级制型的数据。
  - 如果想要提取文本就用text
  - 但是如果你想要提取图片、文件，就要用到content
- 详细一点来讲：
  - 用了request.get方法后，返回一个response对象，这个对象里面存的是服务器返回的所有信息，包括响应头，响应状态码等。
  - 其中返回的网页部分会存在.content和.text两个对象中。如果需要获得这些网页原始数据，我们可以通过r.text 或 r.content来获取数据。
  - .text 存的是.content 编码后的字符串
  - .content中间存的是字节码
  - 一般来说 .text直接用比较方便，返回的是字符串，但是有时候会解析不正常，导致返回的是一堆乱码。这时需要用.content.decode('utf-8')，使其正常显示。

- 总的来说.text是现成的字符串，.content还要编码，但是.text不是所有时候显示都正常（需要用.content.decode()进行手动编码）

1.2 爬取网页的通用代码框架

1.2.1 理解requests库的异常

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请求返回了不成功的状态码
requests.Timeout	请求超时
requests.TooManyRedirects	请求超过了设定的最大重定向次数
requests.URLRequired	URL缺失异常
requests.ConnectTimeout	连接远程服务器超时异常

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

In [ ]:

```
import requests
def getHtmlText(url):
    try:
        r = requests.get(url)
        r.raise_for_status() # 如果状态码不是200，引发HTTPError异常'
        print('====')
        r.encoding=r.apparent_encoding
        return r.text
    except Exception as result:
        return result

if __name__=='__main__':
    url = 'https://www.baidu.com'
#     url = 'www.baidu.com'
    getHtmlText(url)
```

1.3 requests库方法

方法	说明
request()	构造一个请求，支撑以下个方法的基础方法
get	获取网页内容的主要方法
head	获取网页头信息的方法
post	提交post请求的方法
put	提交put请求的方法

方法	说明
patch	提交局部修改请求
delete	提交删除请求

- requests.request(method,url,**kages)
  - method:就是上面的6种方法名
  - **kwags：控制访问的参数
    - params=:字典或字节序列，作为参数增加到url中。
      - 如：{'key1':'value1','key2':'value2'}
      - 或{'key1':'value1','key2':['value2','value3']}
    - data=：字典、字节序列或文件对象，作为request的内容。
      - 如：字典{'key1': 'value1', 'key2': 'value2'}
      - 或 元组 (('key1', 'value1'), ('key1', 'value2'))
    - json=：json格式的数据，作为request的内容。常用使用键值对。
    - headers=：字典，http定制头
    - cookies=：字典或CookieJar
    - auth=：元组，支持http认证功能
    - files=：字典，传输文件：如{'file': open('report.xls', 'rb')}
    - timeout=：设置超时时间，以秒为单位
    - proxies=：字典类型，设定访问代理服务器，可以增加登录认证。
      - 如：{'http': 'http://user:pass@10.10.10.1:1234', 'https': 'https://10.10.10.1:4321'}
    - 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

1.4 应用：爬取京东商品信息

In [ ]:

```
import requests
url = 'https://item.jd.com/65539988138.html'
try:
    r = requests.get(url)
    r.raise_for_status()
    r.encoding=r.apparent_encoding
    print(r.text[:500]) #只取前1000个字符
except Exception as result:
    print( result)
```

1.5 应用：爬取亚马逊商品信息

In [ ]:

```
# 版本1 - 报错：'User-Agent': 'python-requests/2.22.0'
url = 'https://www.amazon.cn/dp/B071R4P6QG'
r = requests.get(url)
r.encoding=r.apparent_encoding
print(r.status_code)
print(r.headers) # 返回的头部信息
print(r.request.headers) # 请求的头部信息
```

In [ ]:

```
# 版本2 - 可正常获取
url = 'https://www.amazon.cn/dp/B071R4P6QG'
kv = {'user-agent': 'Mozilla/5.0'}
r = requests.get(url, headers=kv)
r.encoding = r.apparent_encoding
print(r.status_code)
print(r.request.headers)  # 请求的头部信息
print(r.text)
```

In [ ]:

```
# 标准框架版本
import requests
url = 'https://www.amazon.cn/dp/B071R4P6QG'
kv = {'user-agent': 'Mozilla/5.0'}
try:
    r = requests.get(url, headers=kv)
    r.raise_for_status()
    r.encoding=r.apparent_encoding
    print(r.text[15000:18000]) #取1000-2000个字符
except Exception as result:
    print( result)
```

1.6 百度360搜索关键词提交

In [ ]:

```
import requests
url = 'https://www.so.com/s'
kv = {'user-agent': 'Mozilla/5.0'}
payload ={'q':'加多宝'}
try:
    r=requests.get(url,params=payload,headers=kv)
    print(r.request.headers)
    print(r.request.url)
    r.raise_for_status()
    r.encoding=r.apparent_encoding
    print(r.text[:2000])
except Exception as result:
    print('出错啦',result)
```

1.7 网络图片的爬取和保存

In [ ]:

```
# 版本1：不标准
import requests
path = 'C:/Users/Administrator/Desktop/abc.jpg'
url = 'https://wx2.sinaimg.cn/large/ae2f492fgy1g0pl05djpoj20hz0b0gmm.jpg'
r= requests.get(url)
print(r.status_code)
with open(path,'wb') as f:
    f.write(r.content)
```

In [ ]:

```
# 版本2：标准写法
import requests
import os

url = 'https://wx2.sinaimg.cn/large/ae2f492fgy1g0pl05djpoj20hz0b0gmm.jpg'
kv = {'user-agent': 'Mozilla/5.0'}
root = 'C:/Users/Administrator/Desktop/'
path = root + url.split('/')[-1]
try:
    if not os.path.exists(root): #判断目录是否存在
        os.mkdir(root)
    if not os.path.exists(path): # 判断文件是否已经存在
        r = requests.get(url, headers=kv)
        r.raise_for_status()
        with open(path, 'wb') as f:
            f.write(r.content)
            print('文件保存成功')
    else:
        print('文件已存在')
except:
    print('爬取失败')
```

1.8 IP归属地的查询

In [ ]:

```
# 版本1：不标准
import requests
url='http://ip138.com/iplookup.asp?ip=110.121.3.3'
r = requests.get(url)
print(r.status_code)
r.encoding=r.apparent_encoding
print(r.text[5000:])
```

In [ ]:

```
import requests
url='http://ip138.com/iplookup.asp?ip=110.121.3.3'
try:
    r=requests.get(url)
    r.raise_for_status()
    r.encoding=r.apparent_encoding
    print(r.text[5000:])
except:
    print('爬取失败')
```

2 Beautiful Soup库

2.1 bs4库的基本元素

基本元素	说明
Tag	标签，分别用<></>标明开头和结尾
Name	标签的名字，如<\p>...</p>的名字是'p',格式<tag>.name
Attriutes	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，<>...</>中的字符串，格式：<tag>.string
Comment	标签内字符串的注释部分

In [ ]:

```
# 语法1：from bs4 import BeautifulSoup 从bs4库中引入BeautifulSoup类
# 语法2：import bs4
from bs4 import BeautifulSoup
import requests
r = requests.get('https://www.eyuyun.com/sitemap.xml')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
result=BeautifulSoup(mk,'xml')
print(result)
```

In [101]:

```
# https://python123.io/ws/demo.html
from bs4 import BeautifulSoup
import requests
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
soup=BeautifulSoup(mk,'html.parser')
print(soup.title) # 打印title标签
print(soup.a) # 打印a标签,但只取第一个a标签
print(soup.a.name,type(soup.a.name)) # 打印a标签名字
print(soup.a.parent.name) # 打印a标签父标签名字
print(soup.a.parent.parent.name) # 打印a标签父标签的父标签名字

print(soup.a.attrs)# 打印a标签的属性，以字典形式呈现
print(soup.a.attrs['href']) # 打印a标签href属性的值
print(soup.a.attrs['class']) # 打印a标签class属性的值,以列表形式呈现

print(soup.a.string,type(soup.a.string)) # 获取a标签内的非标签字符串
```

```
200
<title>This is a python demo page</title>
<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic P
ython</a>
a <class 'str'>
p
body
{'href': 'http://www.icourse163.org/course/BIT-268001', 'class': ['py1'], 'id': 'lin
k1'}
http://www.icourse163.org/course/BIT-268001 (http://www.icourse163.org/course/BIT-26
8001)
['py1']
Basic Python <class 'bs4.element.NavigableString'>
```

In [ ]:

```
from bs4 import BeautifulSoup
newsoup = BeautifulSoup('<p><!-- this is a comment --></p><b>this is a commet2</b>','html.parser')
print(newsoup.p.string,type(newsoup.p.string)) # 获取注释内容
print(newsoup.b.string,type(newsoup.b.string))
```

2.2 bs4库html元素的遍历方法

2.2.1 标签树的下行遍历

属性	说明
.contents	子节点的列表，将<\tag>的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

In [96]:

```
from bs4 import BeautifulSoup
import requests
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
soup=BeautifulSoup(mk,'html.parser')
# 取body节点字符串
print(type(soup.body))
print(soup.body)
# 取body儿子节点 - 返回列表：特殊字符也会纳入其中，比如换行符\n
print(soup.body.contents)
# 取列表元素个数
print(len(soup.body.contents))

for child in soup.body.children: # 循环遍历儿子节点
    print(child)

print('================')

for child in soup.body.descendants: # 循环遍历子孙节点
    print(child)
```

```
200
<class 'bs4.element.Tag'>
<body>
<p class="title"><b>The demo python introduces several python courses.</b></p>
<p class="course">Python is a wonderful general-purpose programming language. Y
ou can learn Python from novice to professional by tracking the following cours
es:
<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Ba
sic Python</a> and <a class="py2" href="http://www.icourse163.org/course/BIT-10
01870001" id="link2">Advanced Python</a>.</p>
</body>
['\n', <p class="title"><b>The demo python introduces several python courses.</
b></p>, '\n', <p class="course">Python is a wonderful general-purpose programmi
ng language. You can learn Python from novice to professional by tracking the f
ollowing courses:
<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Ba
sic Python</a> and <a class="py2" href="http://www.icourse163.org/course/BIT-10
01870001" id="link2">Advanced Python</a>.</p>, '\n']
5
```

2.2.2 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

In [ ]:

```
from bs4 import BeautifulSoup
import requests
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text

soup=BeautifulSoup(mk,'html.parser')
print(soup.a.parent) #返回包括父标签在内的所有内容
print('======')
print(soup.a.parent.name) #返回包括父标签名称
print('======')
# 遍历所有上行标签名称
for parent in soup.a.parents:
    if parent is None:
        print(1,parent)
    else:
        print(2,parent.name)
```


2.2.3 标签树的平行遍历

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

- 所有的标签平行遍历，必须发生在处于一个父节点中

In [93]:

```
from bs4 import BeautifulSoup
import requests
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
soup=BeautifulSoup(mk,'html.parser')
```

```
200
<class 'bs4.element.NavigableString'>
 and
Python is a wonderful general-purpose programming language. You can learn Python fro
m novice to professional by tracking the following courses:

None
```

In [95]:

```
from bs4 import BeautifulSoup
import requests
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
soup=BeautifulSoup(mk,'html.parser')
# 后续平行遍历
for sibling in soup.a.next_siblings:
    print(sibling)
print('============')

# 前续平行遍历
for sibling in soup.a.previous_siblings:
    print(sibling)
```

```
200
<class 'bs4.BeautifulSoup'>
 and
<a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Adv
anced Python</a>
.
============
Python is a wonderful general-purpose programming language. You can learn Python fro
m novice to professional by tracking the following courses:
```

2.3 bs4库的html格式化和编码

2.3.1 prettify()方法

In [ ]:

```
# prettify()格式化
from bs4 import BeautifulSoup
import requests
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
soup=BeautifulSoup(mk,'html.parser')

# 对html文本进行格式化
print(soup.prettify())
print('=============')
# 也可以对标签进行处理
print(soup.a.prettify())
```

In [ ]:

```
# prettify()编码utf-8，可以很好地支持中文等其他语言
soup=BeautifulSoup('<p>中文</p>','html.parser')
print(soup.p.sting)
print(soup.p.prettify())
```

2.4 bs4库的html内容查找方法

2.4.1 find_all()

- return = soup.find_all(name,attrs,recursive,string,**kwargs)
  - return:返回一个列表类型，存储查找的结果
  - name:标签名称
  - attrs:对标签属性值的检索字符串，可标注属性检索
  - recursive:是否对所有子孙标签进行查找，默认True
  - string:<>...</>中字符串区域的检索字符串

In [ ]:

```
from bs4 import BeautifulSoup
import requests
import re
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
soup = BeautifulSoup(mk, 'html.parser')

# 知识点1:
result = soup.find_all('a')
print(result)  # 返回一个列表

for link in result:
    print(link)
    print(link.get('href'))  # 获取href属性的值

# 知识点2:
result = soup.find_all(['a', 'b'])  # 查找多个标签
# print(result)

# 知识点3:
for tag in soup.find_all(True):  # 查找所有标签名称
    print(tag.name)

# 知识点4:通过正则,查找含有某字符的标签
for tag in soup.find_all(re.compile('b')):
    print(tag.name)

# 知识点5:搜索带有某属性的某标签
result = soup.find_all('p', 'course')
# print(result)

# 知识点6:指定查找某属性等于某值得标签元素,如id='link1'
result = soup.find_all(id='link1')
# print(result)

# 知识点7:是否查找所有子孙标签。默认True。False代表只对儿子标签进行查找
result = soup.find_all('a', recursive=False)
# print(result)

# 知识点8:检索字符串区域含有指定字符的标签
result = soup.find_all(string='Basic Python') #必须精确输入字符串才可以查找到
print(result)
result = soup.find_all(string=re.compile('Python')) # 可以加入正则表达式
print(result)
```

2.4.1.1 find_all()缩写

- <tag>(...) 等价于 <tag>.find_all(...)
- soup(...) 等价于 soup.find_all(...)

In [ ]:

```
from bs4 import BeautifulSoup
import requests
import re
r = requests.get('https://python123.io/ws/demo.html')
print(r.status_code)
r.encoding = r.apparent_encoding
mk = r.text
soup = BeautifulSoup(mk, 'html.parser')

# soup(...) 等价于 soup.find_all(...)
result = soup('a')
print(result)  # 返回一个列表

<tag>(...) 等价于 <tag>.find_all(...)
result = soup.p('b')
print(result)  # 返回一个列表
```

2.4.2 find拓展方法

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()	在先辈节点中搜索，返回字符串类型，同find_all()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()	在后续平行节点中搜索，返回字符串类型，同find_all()参数
.find_previous_siblings()	在前续平行节点中搜索，返回列表类型，同find_all()参数
.find_previous_sibling()	在前续平行节点中搜索，返回字符串类型 ，同find_all()参数

2.5 应用：大学排名爬取

In [ ]:

```
'''
步骤:
1.从网络获取大学排名的网页内容 getHTMLText()
2.提取网页内容中信息到合适的数据结构 fillUnivList()
3.利用数据结构展示并输出结果 printUnivList()
'''
import requests
import bs4
from bs4 import BeautifulSoup


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


def fillUnivList(ulist, html):
    soup = BeautifulSoup(html, 'html.parser')
#     遍历tbody标签中的tr标签
    for tr in soup.tbody.children:
        if isinstance(tr, bs4.element.Tag): #isinstance() 函数来判断一个对象是否是一个已知的类型
            tds = tr('td')  # 搜索tr标签中的td标签，并返回一个列表
            ulist.append([tds[0].string, tds[1].string, tds[2].string])


def printUnivList(ulist, num):
    print(f'排名\t学校名称\t省市')
    for i in range(num): # rang(num),创建一个0-num的列表
        u = ulist[i]
        print(f'{u[0]}\t{u[1]}\t{u[2]}')


def main():
    uinfo = []
    url = 'http://www.zuihaodaxue.com/zuihaodaxuepaiming2019.html'
    html = getHTMLText(url)
    fillUnivList(uinfo, html)
    printUnivList(uinfo, 20)  # 打印20所大学


main()
```

3 re库

3.1 常用正则操作符

操作符	说明	实例

操作符	说明	实例
.	表示任意单个字符	
[ ]	字符集，对单个字符给出取值范围	[abc]表示a或b或c，[a-z]表示a到z的任意单个字符
[^]	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次或无限次拓展	abc*表示ab、abc、abcc、abccc等
+	前一个字符1次或无限次拓展	abc+表示abc、abcc、abccc等
?	前一个字符0次或1次拓展	abc?表示ab、abc
|	左右表达式任意一个	abc|def表示abc或def
{m}	拓展前一个字符m次	ab{2}c表示abbc
{m:n}	拓展前一个字符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或def
\d	数字，等价于[0-9]	
\w	单词字符，等价于[A-Za-z0-9]	

3.2 正则表达式类型

3.2.1 raw string类型（原生字符串类型）

- re库采用raw string类型表示正则表达式，表示为：r'text'。例如：r'[1-9]\d{5}'

3.2.2 re库主要功能函数

3.2.2.1 re.search()

- 扫描整个字符串并返回第一个成功的匹配。匹配成功re.search方法返回一个匹配的match对象，否则返回None。
- 语法：re.search(pattern, string, flags=0)
  - pattern：正则表达式的字符串或原生字符串表示
  - string：待匹配字符串
  - flags：正则表达式使用时的控制标记
    - re.I 忽略正则表达式的大小写，[A-Z]能够匹配小写字符
    - re.M 正则表达式中的^操作符能够将给定字符串的每行当做匹配开始
    - re.S 正则表达式的.操作符能够匹配所有字符（除换行符外）

In [ ]:

```
import re
match = re.search(r'[1-9]\d{5}','BIT 100081 BIT 12345')
if match:
    print(match.group())
```

3.2.2.2 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。
- 语法：re.match(pattern, string, flags=0)

In [ ]:

```
import re
match = re.match(r'[1-9]\d{5}','BIT 100081 BIT 12345')
if match:
    print(match.group())
else:
    print('match为None')
```

In [ ]:

```
import re
match = re.match(r'[1-9]\d{5}','100081 BIT BIT 12345')
if match:
    print(match.group())
else:
    print('match为None')
```

3.2.2.3 re.match() 与 re.search()的区别

- re.match只匹配字符串的开始，如果字符串开始不符合正则表达式，则匹配失败，函数返回None；而re.search匹配整个字符串，直到找到一个匹配。

In [ ]:

```
import re
print(re.search('www', 'www.runoob.com'))  # 在起始位置匹配
print(re.search('com', 'www.runoob.com').span())
```

3.2.2.4 re.findall()

- 在字符串中找到正则表达式所匹配的所有子串，并返回一个列表，如果没有找到匹配的，则返回空列表。
  - 注意： match 和 search 是匹配一次 findall 匹配所有。
- 语法一：list = re.findall(pattern, string, flags=0)
- 语法二：findall(string[, pos[, endpos]])
  - string : 待匹配的字符串。

- pos : 可选参数，指定字符串的起始位置，默认为 0。
- endpos : 可选参数，指定字符串的结束位置，默认为字符串的长度。

In [ ]:

```
# 语法一：
import re
list = re.findall(r'[1-9]\d{5}','100081 BIT BIT 129345')
if list:
    print(list)
else:
    print('match为None')
```

In [ ]:

```
# 语法二：
import re

pattern = re.compile(r'\d+')   # 查找数字
result1 = pattern.findall('runoob 123 google 456')
result2 = pattern.findall('run88oob123google456', 0, 10)

print(result1)
print(result2)
```

3.2.2.5 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。
- 语法：re.split(pattern, string, maxsplit=0, flags=0)
  - pattern：匹配的正则表达式
  - string：要匹配的字符串。
  - maxsplit：最大分隔次数，剩余部分作为最后一个元素输出。maxsplit=1 分隔一次，默认为 0，不限制次数。可空
  - flags：标志位，用于控制正则表达式的匹配方式。可空

In [ ]:

```
import re
list = re.split(r'[1-9]\d{5}','100081 BIT  129345BIT',maxsplit=1)
if list:
    print(list)
else:
    print('match为None')
```

3.2.2.6 re.finditer()

- 和 findall 类似，在字符串中找到正则表达式所匹配的所有子串，并把它们作为一个迭代器返回，每个迭代元素是match对象。
- 语法：re.finditer(pattern, string, flags=0)

In [7]:

```
import re
for m in re.finditer(r'[1-9]\d{5}','100081 BIT  129345BIT'):
    if m:
        print(m.group())
```

```
100081
129345
```

3.2.2.7 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。
- 语法：re.sub(pattern, repl, string, count=0, flags=0)
  - pattern : 正则中的模式字符串。
  - repl : 替换的字符串，也可为一个函数。
  - string : 要被查找替换的原始字符串。
  - count : 模式匹配后替换的最大次数，默认 0 表示替换所有的匹配。
  - flags : 正则表达式使用时的控制标记

In [ ]:

```
import re
str1 = re.sub(r'[1-9]\d{5}','11111','100081 BIT  129345BIT')
print(str1)
```

3.2.3 re库功能函数的等价用法

In [ ]:

```
# 上述函数都可用该方法
# 原方法：
import re
str1 = re.sub(r'[1-9]\d{5}','11111','100081 BIT  129345BIT')
print(str1)

# 等价方法
pa = re.compile(r'[1-9]\d{5}')
str1 = pa.sub('11111','100081 BIT  129345BIT')
print(str1)
```

3.2.3.1 re.compile()

- compile 函数用于编译正则表达式，生成一个正则表达式（ Pattern ）对象。
- 语法：regex = re.compile(pattern, flags=0)

3.2.4 正则表达式match对象

- 上面讲述的方法中，re.search()、re.match()、re.finditer()返回的是match对象

In [ ]:

```
# 例子
import re
match = re.search(r'[1-9]\d{5}','BIT 100081 BIT 12345')
if match:
    print(match.group())
print(type(match))
```

3.2.4.1 match对象的属性

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

In [ ]:

```
import re
match = re.search(r'[1-9]\d{5}','BIT 100081 BIT 12345')
print(match.string)
print(match.re)
print(match.pos)
print(match.endpos)
```

3.2.4.2 match对象的方法

方法	说明
.group(0)	获取匹配后的字符串
.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())

In [ ]:

```
import re
match = re.search(r'[1-9]\d{5}','BIT 100081 BIT 12345')
print(match.group(0))
print(match.start())
print(match.end())
print(match.span())
```

3.2.5 re库的贪婪匹配和最小匹配

In [ ]:

```
'''
贪婪匹配：re库默认采用贪婪匹配，即输出匹配最长的子串
'''
import re
match = re.search(r'PY.*N','PYANBNCNDN')
print(match.group(0))
```

In [ ]:

```
'''
最小匹配：如何输出最短的子串呢？
'''
import re
match = re.search(r'PY.*?N','PYANBNCNDN')
print(match.group(0))
```

- 最小匹配操作符
  - *? 前一个字符0次或无限次拓展，最小匹配
  - +? 前一个字符1次或无限次拓展，最小匹配
  - ?? 前一个字符0次或1次拓展，最小匹配
  - {m,n}? 拓展前一个字符m至n次（含n次），最小匹配

4 应用：淘宝商品信息爬取

- 目标：获取淘宝搜索页面的信息，提取其中的商品名称和价格
- 理解：
  - 淘宝的搜索接口
  - 翻页的处理
- 程序结构：
  1. 提交商品搜索请求，循环获取页面
  2. 对于每个页面，提取商品名称和价格信息
  3. 将信息输入到屏幕上

In [ ]:

```
# 先构想几个函数和变量，写出框架
# 对每一个函数进行填写

import requests
import re


def GetHtmlText(url, user_agent, cookie):
    try:
        r = requests.get(url, cookies=cookie, headers=user_agent, timeout=30)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        return r.text
    except Exception as result:
        print('访问错误:', result)
        return '访问错误'


def parsePage(ulist, html):
    try:
        plist = re.findall('view_price\"\:\"[\d\.]*\"', html)
        tlist = re.findall('raw_title\"\:\".*?\"', html)
        for i in range(len(plist)):
            price = eval(plist[i].split(':')[1])
            title = eval(tlist[i].split(':')[1])
            ulist.append([title, price])
    except:
        print('')


def printGoodsList(ulist):
    tplt = '{:4}\t{:8}\t{:16}'
    print(tplt.format('Num', 'Price', 'GoodsName'))
    count = 0
    for g in ulist:
        count += 1
        print(tplt.format(count, g[1], g[0]))


def main():
    goods = '书包'
    depth = 2
    start_url = 'https://s.taobao.com/search?q='+goods
    infoList = []

#     headers、cookies必须是字典
    user_agent = {'user-agent': 'Mozilla/5.0'}
    cookie_text = '登录淘宝后，F12查看淘宝cookies,复制进来'
    cookie = {}
    for i in cookie_text.split('; '):
        name, value = i.strip().split("=", 1)
        cookie[name] = value

    for i in range(depth):
        try:
            #             url = 'https://s.taobao.com/search?q=%E4%B9%A6%E5%8C%85&imgfile=&js=1&s
            url = start_url+"&s="+str(44*i)
            html = GetHtmlText(url, user_agent, cookie)
            parsePage(infoList, html)
```

```
        except:
            continue
    printGoodsList(infoList)


main()
```

5 股票数据定向爬虫

- 目标：获取深交所、上交所所有股票的名称和交易
- 输出：保存到文件中
- 网站选择：
  - 原则：股票信息静态存在于html页面中，非js生成，没有robots限制
  - 方法：F12、源代码查看等

In [58]:

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


def GetHtmlText(url, header):
    '''获取网页数据的函数'''
    try:
        r = requests.get(url, headers=header, timeout=30)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        return r.text
    except Exception as result:
        print('访问错误：', result)


def parseHtml(ulist, html):
    '''提取返回html文本数据，写入列表的函数'''
    soup = BeautifulSoup(html, 'html.parser')
    tr = soup.find_all('tr')
    a = 0
    for i in tr:
        a += 1
        # 每一页第一个i，不是我们需要的数据，所以通过判断进行跳过
        if a != 1:
            stock_list = []  # 每开始爬取某个股票信息前，创建一个空列表用来装这个股票的数据
            for j in i:
                if j.string != None and j.string != '\n':
                    stock_list.append(j.string)
    #         print(stock_list)
            if len(stock_list) > 0:
                ulist.append([stock_list[0], stock_list[1], stock_list[2]])
    #         print('==============')


def printInfo(ulist):
    '''将爬取的数据写入文件'''
    try:
        with open('test.txt', 'w') as f:
            f.truncate()  # 先清空文件
        f.close()
        with open('test.txt', 'a') as f:  # 向文件中追加数据
            count = 0
            f.write(f'序号\t股票代码\t股票名称\t最新价格\n')
            for i in ulist:
                count += 1
                f.write(f'{count:4}\t{i[0]:4}\t{i[1]:4}\t{i[2]:4}\n')
        f.close()

    except Exception as result:
        print('写入错误', result)


def main():
    header = {'Mozilla': '5.0'}
    start_url = 'http://app.finance.ifeng.com/list/stock.php?t=hs&f=chg_pct&o=desc'
    depth = 5  # 爬取2页的数据，数字可自定义
    ulist = []
```

```
    for i in range(depth):
        url = start_url + "&p=" + str(i)
        html = GetHtmlText(url, header)
        parseHtml(ulist, html)
        printInfo(ulist)
    print('任务完成')


main()
```

任务完成