Xpath处理xml文档

术语：

在Xpath中有七种类型的节点：元素、属性、文本、命名空间、处理指令、注释、以及根节点．XML是被作为节点树来对待的．树的根被称为文档节点或根节点．

有以下XML文档：

```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

以上XML文档中的节点例子：

```
<bookstore> (文档节点)
<author>J K. Rowling</author> (元素节点)
lang="en" (属性节点)
```

基本值：无父或无子的节点
如：

```
J K. Rowling
"en"
```

节点关系：

父:每个元素以及属性都有一个父．在本例中book 元素是 title、author、year 以及 price 元素的父．

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

子：元素节点可有零个、一个或多个子．在本例中title、author、year 以及 price 元素都是 book 元素的子．

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

同胞：拥有相同的父的节点．在本例中title、author、year 以及 price 元素都是同胞

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

先辈:某节点的父、父的父 等等．在本例中title元素的先辈是book元素和 bookstore元素．

```
<bookstore>
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
</bookstore>
```

后代：
某个节点的子 子的子 等等.在本例中bookstore 的后代是 book、title、author、year 以及price元素．

```
<bookstore>
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
</bookstore>
```

二.语法：

In [15]:
```
x = """<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book>
  <title lang="eng">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="ing">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
"""
```

2.1 nodename 选取此节点的所有子节点

In [10]:
```
from lxml import etree
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('book')
```

Out[10]:
```
[<Element book at 0x110a26308>, <Element book at 0x111345dc8>]
```

2.2 / 选取根元素

In [3]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('/bookstore')
```

Out[3]:
```
[<Element bookstore at 0x110b962c8>]
```

In [5]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('book/title')
```

Out[5]:
```
[<Element title at 0x110b2a4c8>, <Element title at 0x1114ab188>]
```

2.3 // 获取所有book子元素而不管他在文档中的位置

In [6]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('//book')
```

Out[6]:
```
[<Element book at 0x1113bb9c8>, <Element book at 0x1114ab608>]
```

2.4 //@lang 选取名为lang的所有属性

In [7]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('//@lang')
```

Out[7]:
```
['eng', 'eng']
```

三. 谓语
谓语用来查找某个特定的节点或者包含某个指定的值的节点，谓语被嵌在方括号中. 在下面的表格中，我们列出了带有谓语的一些路径表达式，以及表达式的结果.

3.1 选取属于bookstore子元素的第一个book元素

In [8]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('/bookstore/book[1]')
```

Out[8]:
```
[<Element book at 0x110b2a348>]
```

3.2 选取属于bookstore子元素的最后一个book元素

In [10]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('/bookstore/book[last()]')
```

Out[10]:
```
[<Element book at 0x110b2a548>]
```

3.3 选取属于bookstore子元素的倒数第二个book元素

In [12]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('/bookstore/book[last()-1]')
```

Out[12]:
```
[<Element book at 0x110b2a588>]
```

3.4 选取最前面的两个属于bookstore元素的子元素的book元素

In [13]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('/bookstore/book[position()<3]')
```

Out[13]:
```
[<Element book at 0x110b2a488>, <Element book at 0x1114abc08>]
```

3.5 选取所有拥有名为lang的属性的title元素

In [14]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('//title[@lang]')
```

Out[14]:
```
[<Element title at 0x110b2a748>, <Element title at 0x110bb7448>]
```

3.6 选取所有title元素，且这些元素拥有值为eng的lang属性

In [20]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath("//title[@lang='ing']")
```

Out[20]:
```
[<Element title at 0x1114ab8c8>]
```

3.7 选取bookstore元素的所有book元素，且其中的price元素的值须大于35.00

In [21]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath("/bookstore/book[price>35.00]")
```

Out[21]:
```
[<Element book at 0x1114ab9c8>]
```

3.8 选取bookstore元素中的book元素的所有title元素且其中的price元素的值须大于35.00.

In [22]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath("/bookstore/book[price>35.00]/title")
```

Out[22]:
```
[<Element title at 0x110b9a588>]
```

四. 通配符
4.1 * 匹配任何元素节点

In [24]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('/bookstore/*')
```

Out[24]:
```
[<Element book at 0x1114b7808>, <Element book at 0x1114b7b08>]
```

4.2 @*匹配任何属性节点

In [23]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('//title[@*]')
```

Out[23]:
```
[<Element title at 0x1114aba08>, <Element title at 0x1114abac8>]
```

五. 选取若干路径

5.1 选取book元素的所有title和price元素

In [25]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath('//book/title | //book/price')
```

Out[25]:
```
[<Element title at 0x1114ab0c8>,
 <Element price at 0x1114b71c8>,
 <Element title at 0x1114b7d08>,
 <Element price at 0x1114b7d88>]
```

5.2 选取文档中的所有title和price元素

In [26]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath("//title | //price")
```

Out[26]:
```
[<Element title at 0x107182348>,
 <Element price at 0x1114ab788>,
 <Element title at 0x1114b7948>,
 <Element price at 0x1114b7e08>]
```

5.3 选取属于bookstore元素的book元素的所有title元素，以及文档中所有的price元素

In [27]:
```
root = etree.fromstring(bytes(x, encoding="utf-8"))
root.xpath("/bookstore/book/title | //price")
```

Out[27]:
```
[<Element title at 0x1114b7548>,
 <Element price at 0x1114b7f08>,
 <Element title at 0x1114be108>,
 <Element price at 0x1114be288>]
```

示例代码:(拉取福彩双色球中奖号码)

In [3]:
```
from lxml import etree
import requests
url = "http://kaijiang.zhcw.com/zhcw/html/ssq/list_1.html"
headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"}
result = []
resp = requests.get(url=url, headers=headers)
html = etree.HTML(resp.text)
y = html.xpath('//tr/td')
for c in range(0, len(y), 7):
    tmp = []
    group = html.xpath('//tr/td')[c:c+7]
    if len(group) != 7:
        continue
    for (k, v) in enumerate(group):
        if k < 2:
            tmp.append(v.text)
        elif k == 2:
            w = v.xpath('em')
            for j in w:
                tmp.append(j.text)
        else:
            s = v.xpath('strong')
            for j in s:
                tmp.append(j.text)
    result.append(tmp)
print(result[0:3])
```

```
[['2019-08-08', '2019092', '09', '17', '27', '28', '32', '33', '08', '325,113,036', '6', '67'], ['2019-08-06', '2019091', '07', '10', '21', '24', '29', '32', '11',
'321,773,688', '2', '94'], ['2019-08-04', '2019090', '02', '03', '06', '08', '14', '22', '04', '355,808,718', '2', '93']]
```