Python 程序设计教案

<table>
<tr><td>本次授课内容</td><td colspan="2">13.1 HTML 和 JavaScript 基础
13.2 urllib 基本应用与爬虫案例
13.3 scrapy 爬虫案例</td></tr>
<tr><td>本次课的
教学目的</td><td colspan="2">了解常用的 HTML 标签
了解在网页中使用 JavaScript 代码的几种方式
掌握 Python 标准库 urllib 的用法
掌握 Python 扩展库 scrapy 的用法</td></tr>
<tr><td>本次课教学
重点与难点</td><td colspan="2">urllib 应用
scrapy 应用</td></tr>
<tr><td>教学方法
教学手段</td><td colspan="2">PPT、边讲边练</td></tr>
<tr><td rowspan="3">课堂教学
时间分配</td><td>教学内容</td><td>时间分配（分）</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>课堂教学设计</td><td colspan="2">首先介绍 HTML 和 JavaScript 基本语法和概念，然后介绍标准库 urllib 和扩展库 scrapy 的基本用法，最后通过几个案例演示网络爬虫程序的编写。</td></tr>
<tr><td>实　验</td><td colspan="2">教材中前三节的例题。</td></tr>
<tr><td>思考题及作业题</td><td colspan="2">本章所有课后习题。</td></tr>
<tr><td>备　注</td><td colspan="2"></td></tr>
<tr><td>教学后记</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">第　一　节　课</td></tr>
</table>

课堂重点内容详解	**13.1.1 HTML 基础**
	
	（1）h 标签
	
	`<h1>一级标题</h1>`
	`<h2>二级标题</h2>`
	`<h3>三级标题</h3>`
	
	（2）p 标签
	
	`<p>这是一个段落</p>`
	
	（3）a 标签
	
	`<a href="http://www.baidu.com">点这里</a>`
	
	（4）img 标签
	
	`<img src="Python 可以这样学.jpg" width="200" height="300" />`
	`<img src="http://www.tup.tsinghua.edu.cn/upload/bigbookimg/072406-01.jpg" width="200" height="300" />`
	
	（5）table、tr、td 标签
	
	`<table border="1">`
	`<tr>`
	`<td>第一行第一列</td>`
	`<td>第一行第二列</td>`
	`</tr>`
	`<tr>`
	`<td>第二行第一列</td>`
	`<td>第二行第二列</td>`
	`</tr>`
	`</table>`
	
	（6）ul、ol、li
	
	`<ul id="colors" name="myColor">`
	`<li>红色</li>`
	`<li>绿色</li>`
	`<li>蓝色</li>`
	`</ul>`
	
	（7）div 标签
	
	`<div id="yellowDiv" style="background-color:yellow;border:#FF0000 1px solid;">`
	`<ol>`
	`<li>红色</li>`
	`<li>绿色</li>`
	`<li>蓝色</li>`
	`</ol>`
	`</div>`
	`<div id="reddiv" style="background-color:red">`

```
    <p>第一段</p>
    <p>第二段</p>
</div>
```

13.1.2 JavaScript 基础

（1）在网页中使用 JavaScript 代码的方式

```
<html>
    <body>
        <form>
            <input type="button" value="保存" onClick="alert('保存成
功');">
        </form>
    </body>
</html>
```

```
<html>
    <body>
        <div id="test">静态内容</div>
    </body>
    <script type="text/javascript">
        document.getElementById("test").innerHTML="动态内容";
    </script>
</html>
```

如果一个网站中会用到大量的 JavaScript 代码，一般会把这些代码按功能划分到不同函数中，并把这些函数封装到一个扩展名为 js 的文件中，然后在网页中使用。例如，和网页在同一个文件夹中的 myfunctions.js 内容如下：

```
function modify(){
    document.getElementById("test").innerHTML="动态内容";
}
```

在下面的页面文件中，把外部文件 myfunctions.js 导入，然后调用了其中的函数：

```
<html>
<head>
    <script type="text/javascript" src="myfunctions.js"></script>
</head>
    <body>
        <div id="test">静态内容</div>
    </body>
    <script type="text/javascript">modify();</script>
</html>
```

（2）常用 JavaScript 事件

```
<html>
    <body onLoad="alert('页面开始加载');">
        <div id="test">静态内容</div>
    </body>
</html>
```

除了常用的事件之外，还有一些特殊的方式可以执行 JavaScript 代码。例如，下面的代码演示了在链接标签<a>中使用 href 属性指定 JavaScript 代码的用法。

```
<html>
    <script type="text/javascript">
        function test(){alert('提示信息');}
    </script>
    <body>
        <a href="javascript:test();">点这里</a>
    </body>
</html>
```

第 二 节 课

下面代码为读取并显示 http://www.python.org 页面的内容，具体读取页面的结果不再占用篇幅在此展示。

```
>>> import urllib.request
>>> fp = urllib.request.urlopen(r'http://www.python.org')
>>> print(fp.read(100))                #读取 100 个字节
>>> print(fp.read(100).decode())     #使用 UTF8 进行解码
>>> fp.close()                           #关闭连接
```

下面的代码演示了如何使用 GET 方法读取并显示指定 url 的内容。

```
>>> import urllib.request
>>> import urllib.parse
>>> params = urllib.parse.urlencode({'spam': 1, 'eggs': 2, 'bacon': 0})
>>> url = "http://www.musi-cal.com/cgi-bin/query?%s" % params
>>> with urllib.request.urlopen(url) as f:
    print(f.read().decode('utf-8'))
```

下面的代码演示了如何使用 POST 方法提交参数并读取指定页面内容。

```
>>> import urllib.request
>>> import urllib.parse
>>> data = urllib.parse.urlencode({'spam': 1, 'eggs': 2, 'bacon': 0})
>>> data = data.encode('ascii')
>>> with urllib.request.urlopen("http://requestb.in/xrbl82xr",
                                data) as f:
print(f.read().decode('utf-8'))
```

例 13-1 爬取公众号文章中的图片。

```
from re import findall
from urllib.request import urlopen

url = 'https://mp.weixin.qq.com/s?
__biz=MzI4MzM2MDgyMQ==&mid=2247486249&idx=1&sn=a37d079f541b194970428fb2fd7
a1ed4&chksm=eb8aa073dcfd2965f2d48c5ae9341a7f8a1c2ae2c79a68c7d2476d8573c91e1de2e23
7c98534&scene=21#wechat_redirect'
with urlopen(url) as fp:
    content = fp.read().decode()

pattern = 'data-type="png" data-src="(.+?)"'
```

```
#查找所有图片链接地址
result = findall(pattern, content)
#逐个读取图片数据，并写入本地文件
for index, item in enumerate(result):
    with urlopen(str(item)) as fp:
        with open(str(index)+'.png', 'wb') as fp1:
            fp1.write(fp.read())
```

例 13-2 使用 scrapy 框架编写爬虫程序。

```
import os
import urllib.request
import scrapy

class MySpider(scrapy.spiders.Spider):
    #爬虫的名字，每个爬虫必须有不同的名字
    name = 'mySpider'
    allowed_domains = ['www.sdibt.edu.cn']
    #要爬取的起始页面，必须是列表，可以包含多个 url
    start_urls = ['http://www.sdibt.edu.cn/info/1026/11238.htm']

    #对每个要爬取的页面，会自动调用下面这个方法
    def parse(self, response):
        self.downloadWebpage(response)
        self.downloadImages(response)

        #检查页面中的超链接，并继续爬取
        hxs = scrapy.Selector(response)
        sites = hxs.xpath('//ul/li')
        for site in sites:
            link = site.xpath('a/@href').extract()[0]
            if link == '#':
                continue
            #把相对地址转换成绝对地址
            elif link.startswith('..'):
                next_url = os.path.dirname(response.url)
                next_url += '/'+link
            else:
                next_url = link
            #生成 Request 对象，并指定回调函数
            yield scrapy.Request(url=next_url,
                                 callback=self.parse_item)

    #回调函数，对起始页面中的每个超链接起作用
    def parse_item(self, response):
        self.downloadWebpage(response)
        self.downloadImages(response)

    #下载当前页面中所有的图片
    def downloadImages(self, response):
        hxs = scrapy.Selector(response)
```

```
        images = hxs.xpath('//img/@src').extract()
        for image_url in images:
            imageFilename = image_url.split('/')[-1]
            if os.path.exists(imageFilename):
                continue
            #把相对地址转换成绝对地址
            if image_url.startswith('..'):
                image_url = os.path.dirname(response.url)\
                            + '/' + image_url
            #打开网页图片
            fp = urllib.request.urlopen(image_url)
            #创建本地图片文件
            with open(imageFilename,'wb') as f:
                f.write(fp.read())
            fp.close()

    #把网页内容保存为本地文件
    def downloadWebpage(self, response):
        filename = response.url.split('/')[-1]
        with open(filename, 'wb') as f:
            f.write(response.body)
```

例 13-3 使用 scrapy 框架编写爬虫程序，爬取天涯小说。

```
import scrapy

class MySpider(scrapy.spiders.Spider):
    #爬虫的名字，每个爬虫必须有不同的名字
    name = 'spiderYichangGuishi'
    #要爬取的小说首页，第一篇
    start_urls = ['http://bbs.tianya.cn/post-16-1126849-1.shtml']
    #对每个要爬取的页面，会自动调用下面这个方法
    def parse(self, response):
        #用来存放当前页中的小说正文
        content = []
        for i in response.xpath('//div'):
            if i.xpath('@_hostid').extract()==['13357319']:
                for j in i.xpath('div//div'):
                    #提取文本，过滤干扰符号
                    c = j.xpath('text()').extract()
                    g = lambda x:x.strip('\n\r\u3000').replace('<br>',
'\n').replace('|', '')
                    c = '\n'.join(map(g, c)).strip()
                    content.append(c)
        with open('result.txt', 'a+', encoding='utf8') as fp:
            fp.writelines(content)

        url = response.url        #获取下一页网址并继续爬取
        d = url[url.rindex('-')+1:url.rindex('.')]
        u = 'http://bbs.tianya.cn/post-16-1126849-{0}.shtml'
```

```
        next_url = u.format(int(d)+1)
        try:
            yield scrapy.Request(url=next_url,
                                 callback=self.parse)
        except:
            pass
```

Python 程序设计教案

本次授课内容	13.4 BeautifulSoup 用法简介 13.5 requests 基本操作与爬虫案例 13.5 selenium 爬虫案例
本次课的 教学目的	掌握 Python 扩展库 BeautifulSoup4 的用法 掌握 Python 扩展库 requests 的用法 掌握 Python 扩展库 selenium 的用法
本次课教学 重点与难点	扩展库 BeautifulSoup、requests、selenium 的应用。
教学方法 教学手段	PPT、边讲边练
课堂教学 时间分配	**教学内容** / **时间分配（分）** / /
课堂教学设计	首先介绍扩展库的基本语法和操作，然后通过几个案例演示 BeautifulSoup、requests、selenium 的应用。
实　验	课堂上讲解的所有例题。
思考题及作业题	本章所有课后习题。
备　注	
教学后记	

课堂重点内容详解

第 一 节 课

13.4 BeautifulSoup 用法简介

（1）代码补全

大多数浏览器能够容忍一些错误的 HTML 代码，例如某些

闭合的标签，也可以正常渲染和显示。但是如果把读取到的网页源代码直接使用正则表达式进行分析，会出现误差。这个问题可以使用 BeautifulSoup 来解决。在使用给定的文本或网页代码创建 BeautifulSoup 对象时，会自动补全缺失的标签，也可以自动添加必要的标签。

（2）获取指定标签的内容或属性

```
>>> soup.title                                  #访问<title>标签的内容
<title>The Dormouse's story</title>
>>> soup.title.name                             #查看标签的名字
'title'
>>> soup.title.text                             #查看标签的文本
"The Dormouse's story"
>>> soup.title.string                           #查看标签的文本
"The Dormouse's story"
>>> soup.title.parent                           #查看上一级标签
<head><title>The Dormouse's story</title></head>
>>> soup.head
<head><title>The Dormouse's story</title></head>
>>> soup.b                                       #访问<b>标签的内容
<b>The Dormouse's story</b>
>>> soup.body.b                                 #访问<body>中<b>标签的内容
<b>The Dormouse's story</b>
>>> soup.name                    #把整个 BeautifulSoup 对象看作标签对象
'[document]'
>>> soup.body                    #查看 body 标签内容
>>> soup.p                                       #查看段落信息
<p class="title"><b>The Dormouse's story</b></p>
>>> soup.p['class']                             #查看标签属性
['title']
>>> soup.p.get('class')                         #也可以这样查看标签属性
['title']
>>> soup.p.text                                  #查看段落文本
"The Dormouse's story"
>>> soup.p.contents                              #查看段落内容
[<b>The Dormouse's story</b>]
```

```
>>> soup.a
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
>>> soup.a.attrs                          #查看标签所有属性
{'class': ['sister'], 'href': 'http://example.com/elsie', 'id':
'link1'}
>>> soup.find_all('a')                 #查找所有<a>标签
>>> soup.find_all(['a', 'b'])         #同时查找<a>和<b>标签
>>> import re
>>> soup.find_all(href=re.compile("elsie"))
                                      #查找 href 包含特定关键字的标签
[<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>]
>>> soup.find(id='link3')              #查找属性 id='link3'的标签
<a class="sister" href="http://example.com/tillie"
id="link3">Tillie</a>
>>> soup.find_all('a', id='link3')   #查找属性'link3'的 a 标签
[<a class="sister" href="http://example.com/tillie"
id="link3">Tillie</a>]
>>> for link in soup.find_all('a'):
    print(link.text,':',link.get('href'))
>>> print(soup.get_text())                  #返回所有文本
>>> soup.a['id'] = 'test_link1'            #修改标签属性的值
>>> soup.a
<a class="sister" href="http://example.com/elsie"
id="test_link1">Elsie</a>
>>> soup.a.string.replace_with('test_Elsie') #修改标签文本
'Elsie'
>>> soup.a.string
'test_Elsie'
>>> print(soup.prettify())                      #查看修改后的结果
>>> for child in soup.body.children:     #遍历直接子标签
    print(child)
>>> for string in soup.strings:           #遍历所有文本，结果略
    print(string)

>>> test_doc = '<html><head></head><body><p></p><p></p></body></heml>'
>>> s = BeautifulSoup(test_doc, 'lxml')
>>> for child in s.html.children:        #遍历直接子标签
    print(child)
>>> for child in s.html.descendants:     #遍历子孙标签
    print(child)
```

第 二 节 课

例 13-4 使用 requests 库爬取微信公众号“Python 小屋”文章“Python 使用集合实现素数筛选法”中的所有超链接。

```
>>> import requests
>>> url = 'https://mp.weixin.qq.com/s?
__biz=MzI4MzM2MDgyMQ==&mid=2247486531&idx=1&sn=7eeb27a03e2ee8ab4152563b
b110f248&chksm=eb8aa719dcfd2e0f7b1731cfd8aa74114d68facf1809d7cdb0601e3d
3be8fb287cfc035002c6#rd'
>>> r = requests.get(url)
```

```
>>> r.status_code       #响应状态码
200
>>> r.text[:300]         #查看网页源代码前 300 个字符
>>> '筛选法' in r.text  #测试网页源代码中是否包含字符串'筛选法'
True
>>> r.encoding           #查看网页编码格式
>>> links = re.findall(r'<a .*?href="(.+?)"', r.text)
                        #使用正则表达式查找所有超链接地址
>>> for link in links:
    if link.startswith('http'):
        print(link)
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(r.content, 'lxml')
>>> for link in soup.findAll('a'):  #使用 BeautifulSoup 查找超链接地址
    href = link.get('href')
    if href.startswith('http'):      #只输出绝对地址
        print(href)
```

例 13-5 读取并下载指定的 URL 的图片文件。

```
>>> import requests
>>> picUrl = r'https://www.python.org/static/opengraph-icon-
200x200.png'
>>> r = requests.get(picUrl)
>>> r.status_code
>>> with open('pic.png', 'wb') as fp:
    fp.write(r.content)                  #把图像数据写入本地文件
```

例 13-6 使用 selenium 编写爬虫程序，获取指定城市的当前天气。

```
import re
from selenium import webdriver

#指定引擎
driver = webdriver.Edge()
city = input('请输入要查询的城市：').lower()
#获取指定 URL 的信息，并进行渲染
driver.get(r'http://openweathermap.org/find?q={0}'.format(city))
#网页内容渲染结束之后获取网页源代码，并转换成小写
content = driver.page_source.lower()
matchResult = re.search(r'<a href="(.+?)">\s+'+city+'.+?]', content)
if matchResult:
    print(matchResult.group(0))
else:
    print('查不到，请检查城市名字。')
```