《Python 爬虫项目教程(微课版)（第 2 版）》习题参考答案

练习 1

1．Flask 是一个比较简单的 Python Web 开发框架，你知道 Python 还有哪些比较流行的 Web 开发框架吗？说说它们的主要区别。

答：

企业级开发框架——Django

高并发处理框架——Tornado

快速建站的框架——Flask

底层自定义协议网络框架——Twisted

2．使用 GET 方法提交数据与使用 POST 方法提交数据有什么不同？在 Flask 中如何获取提交的数据？

答：

Flask 获取 GET/POST 数据： flask.request.values.get(参数名称，默认值)

3．使用 Flask 编写一个 Web 程序，它接收如下代码提交的 user_name 与 user_pass 数据。

```
<form name="frm" method="post" action="/login">
<input type="text" name="user_name" >
<input type="password" name="user_pass">
<input type="submit" value="Login" >
</form>
```

答：

```
import flask
app=flask.Flask("web")

@app.route("/")
def index():
    s='''
<form name="frm" method="post" action="/login">
<input type="text" name="user_name" >
<input type="password" name="user_pass">
<input type="submit" value="Login" >
</form>
    '''
    return s
```

```
@app.route("/login",methods=["GET","POST"])
def login():
    user_name=flask.request.values.get("user_name","")
    user_pass=flask.request.values.get("user_pass","")
    return user_name+","+user_pass

app.debug=True
app.run()
```

4．说出下面正则表达式匹配的字符串分别是什么。

（1）r"\w+\s"

（2）r"\w+\b"

（3）r"\d+-\d+"

（4）r"\w+@(\w+\.)+\w+"

（5）r"(b|cd)ef"

答： 省略

5 .使用正则表达式匹配 HTML 代码中所有形如<img src="http:// …/文件名.jpg">的 JPG 图像文件，找出一个网站中所有这样表示的 JPG 图像文件，并下载这些图像文件。

答：

```
import urllib.request
import re

def getHtml(url):
    html=""
    try:
        resp=urllib.request.urlopen(url)
        data=resp.read()
        try:
            html=data.decode().lower()
        except:
            html=data.decode("gbk").lower()
    except Exception as err:
        print(err)
    return html

def parseHtml(html):
```

```
    try:
        imgReg=r'<img.+src\s*=\s*"http://.+\.jpg"\s*>'
        srcReg=r'http://.+\.jpg'
        while True:
            m=re.search(imgReg,html)
            if m:
                s=html[m.start():m.end()]
                n=re.search(srcReg,s)
                src=s[n.start():n.end()]
                download(src)
                html=html[m.end():]
            else:
                break
    except Exception as err:
        print(err)

def download(src):
    try:
        print("downloding",src)
        resp=urllib.request.urlopen(src)
        data=resp.read()
        p=src.rfind("/")
        fn=src[p+1:]
        fobj=open(fn,"wb")
        fobj.write(data)
        fobj.close()
        print("downloaded",fn)
    except Exception as err:
        print(err)

url="......"
html=getHtml(url)
parseHtml(html)
```

练习 2

1．简单说明 Beautiful Soup 解析数据的特点。

答： 省略

2．用 Beautiful Soup 装载下面的 HTML 文档，并以规范的格式输出，比较与原来 HTML 文档的区别，说明 Beautiful Soup 是如何修改的。

```
    <body>
    <div>Hi<br>
    <span>Hello</SPAN>
    <p><div>End</div>
```

答：

```
from bs4 import BeautifulSoup

html='''
<body>
<div>Hi<br>
<span>Hello</SPAN>
<p><div>End</div>
'''
soup=BeautifulSoup(html,"html.parser")
print(soup.prettify())
```

3．重新编写本书项目 1 中爬取外汇网站数据的程序，使用 Beautiful Soup 分解出<tr>…</tr>中的<td>…</td>数据。

答：

```
import urllib.request
from bs4 import BeautifulSoup
import sqlite3

class MySpider:
    def openDB(self):
        # 初始化数据库，创建数据库 rates.db 与一张空表 rates
        self.con = sqlite3.connect("rates.db")
        self.cursor = self.con.cursor()
        try:
            self.cursor.execute("drop table rates")
        except:
            pass
```

```
        sql = "create table rates (Currency varchar(256) primary
key,TSP float,CSP float, TBP float, CBP float,Time varchar(256))"
        try:
            self.cursor.execute(sql)
        except:
            pass

    def closeDB(self):
        # 关闭数据库
        self.con.commit()
        self.con.close()

    def insertDB(self, Currency, TSP, CSP, TBP, CBP, Time):
        # 记录插入数据库
        try:
            sql = "insert into rates (Currency,TSP,CSP,TBP,CBP,Time)
values (?,?,?,?,?,?)"
            self.cursor.execute(sql, [Currency, TSP, CSP, TBP, CBP,
Time])
        except Exception as err:
            print(err)

    def show(self):
        # 显示函数
        self.cursor.execute("select Currency,TSP,CSP,TBP,CBP,Time from
rates")
        rows = self.cursor.fetchall()
        print("%-18s%-12s%-12s%-12s%-12s%-12s" % ("Currency", "TSP",
"CSP", "TBP", "CBP", "Time"))
        for row in rows:
            print("%-18s%-12.2f%-12.2f%-12.2f%-12.2f%-12s" % (row[0],
row[1], row[2], row[3], row[4], row[5]))



    def spider(self, url):
        # 爬虫函数
        try:
            resp = urllib.request.urlopen(url)
            data = resp.read()
            html = data.decode()
            soup=BeautifulSoup(html,"html.parser")
            div=soup.find("div",attrs={"id":"realRateInfo"})
```

```
            trs=div.find_all("tr")
            for tr in trs[1:]:
                tds=tr.find_all("td")
                Currency = tds[0].text.strip()
                TSP = float(tds[3].text)
                CSP = float(tds[4].text)
                TBP = float(tds[5].text)
                CBP = float(tds[6].text)
                Time = tds[7].text.strip()
                self.insertDB(Currency, TSP, CSP, TBP, CBP, Time)
        except Exception as err:
            print(err)

    def process(self):
        # 爬取过程
        self.openDB()
        self.spider("http://fx.cmbchina.com/hq/")
        self.show()
        self.closeDB()


# 主程序
spider = MySpider()
spider.process()
```

4．下面是一段 HTML 代码：

```
<body>
<bookstore>
<book id="b1">
  <title lang="english">Harry Potter</title>
  <price>23.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
  <price>39.95</price>
</book>
<book id="b3">
  <title lang="english">Learning Python</title>
  <price>30.20</price>
</book>
</bookstore>
</body></html>
```

试用 Beautiful Soup 完成下面的任务。

（1）找出所有书的名称。

（2）找出所有英文书的名称与价格。

（3）找出价格在 30 元以上的所有书的名称。

答：

```
from bs4 import BeautifulSoup

html='''
<body>
<bookstore>
<book id="b1">
  <title lang="english">Harry Potter</title>
  <price>23.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
  <price>39.95</price>
</book>
<book id="b3">
  <title lang="english">Learning Python</title>
  <price>30.20</price>
</book>
</bookstore>
</body></html>
'''
soup=BeautifulSoup(html,"html.parser")
books=soup.find_all("book")
for b in books:
    print(b.find("title").text)
print()
books=soup.find_all("book")
for b in books:
    title=b.find("title",attrs={"lang":"english"})
    if title:
        print(title.text)
print()
for b in books:
    price=b.find("price").text
    if float(price)<30:
        print(b.find("title").text)
```

5．编写一个爬取 Python 最新版本的 Windows 64 位压缩包的程序，程序分两个部分。

（1）访问 Python 下载页面“https://www.python.org/downloads/”，如图 2-8-1 所示。

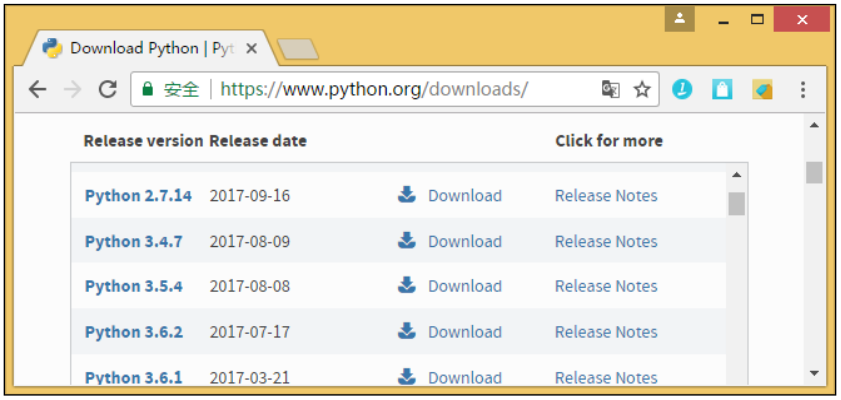


图 2-8-1　Python 下载页面

分析 HTML 代码结构，编写下面的程序，爬取所有发行的 Python 版本与下载地址。

```
    from bs4 import  Beautiful Soup
    import urllib.request
    def searchPython(url):
         resp=urllib.request.urlopen(url)
         data=resp.read()
         html=data.decode()
         soup=Beautiful Soup(html,"lxml")
         ol=soup.find(name="ol",attrs={"class":"list-row-container menu"})
         lis=ol.find_all("li")
         for li in lis:
              a=li.find(name="span",attrs={"class":"release-number"}).
find("a")
              python=a.text
              url=a["href"]
              print("%-20s %s" %(python,url))
    try:
         searchPython("https://www.python.org/downloads/")
    except Exception as e:
         print(e)
```

执行该程序，部分结果如下：

```
    Python 2.5.4          /download/releases/2.5.4/
    Python 2.4.6          /download/releases/2.4.6/
    Python 2.5.3          /download/releases/2.5.3/
    Python 2.6.1          /download/releases/2.6.1/
    Python 3.0.0          /download/releases/3.0/
    ……
```

（2）在这个程序的基础上进一步编写程序，找出最新的版本，自动进入最新版本下载页面，找出这个版本的 Windows 64 位的 ZIP 压缩包的下载地址，并自动下载这个 Python 压缩包。

答：

```
from bs4 import  BeautifulSoup
```

```
import urllib.request
import re

def getLatestVersion(pList):
    python=""
    mVersion=""
    url=""
    for p in pList:
        m=re.search(r"(\d+)\.(\d+)\.(\d+)",p["python"])
        group=m.groups()
        s=""
        for g in group:
            s=s+("%03d"%int(g))
        if s>mVersion:
            python=p["python"]
            url=p["url"]
            mVersion=s
    return python,url

def searchPython(surl):
    resp=urllib.request.urlopen(surl)
    data=resp.read()
    html=data.decode()
    soup=BeautifulSoup(html,"lxml")
    div=soup.find("div",attrs={"class":"row download-list-widget"})
    ol=div.find(name="ol",attrs={"class":"list-row-container menu"})
    lis=ol.find_all("li")
    pList=[]
    for li in lis:
        a=li.find(name="span",attrs={"class":"release-
number"}).find("a")
        python=a.text
        url=urllib.request.urljoin(surl,a["href"])
        print("%-20s %s" %(python,url))
        pList.append({"python":python,"url":url})
    return pList

try:
    pList=searchPython("https://www.python.org/downloads/")
    python,url=getLatestVersion(pList)
    print("The latest version")
    print(python,url)
except Exception as e:
```

```
    print(e)
```

练习 3

1．简单说明 xpath 解析数据的特点。

答：略

2．用 lxml 装载下面的 HTML 文档，并打印出规范的格式，比较与原来 HTML 文档的区别，说明 lxml 是如何修改的？

```
<body>
<div>Hi<br>
<span>Hello
<p><div>End</div>
```

答：

```
from lxml import etree
html='''
<body>
<div>Hi<br>
<span>Hello
<p><div>End</div>
'''
root=etree.HTML(html)
print(etree.tostring(root,pretty_print=True, encoding='unicode',
method='html'))
```

3. 重新使用 xpath 编写爬虫程序，爬取本教程项目二中爬取名人名言数据。

答：

```
import urllib.request
import sqlite3
from lxml import etree

class MySpider:

    def openDB(self):
        #初始化数据库
        self.con = sqlite3.connect("quotes.db")
        self.cursor = self.con.cursor()
        try:
            self.cursor.execute("drop table tags")
        except:
            pass
        try:
```

```
            self.cursor.execute("drop table quotes")
        except:
            pass
        self.cursor.execute("create table quotes (ID INTEGER PRIMARY
KEY AUTOINCREMENT  ,quote varchar(1024),author varchar(256),author_link
varchar(256))")
        self.cursor.execute("create table tags (ID INTEGER PRIMARY KEY
AUTOINCREMENT ,qID int,tag varchar(256),tag_link varchar(256))")

    def insertQuote(self,quote,author,author_link):
        sql="insert into quotes (quote,author,author_link) values
(?,?,?)"
        self.cursor.execute(sql,[quote,author,author_link])
        self.cursor.execute("select ID from quotes order by ID desc
limit 1")
        row=self.cursor.fetchone()
        ID=row[0]
        return ID

    def insertTag(self,qID,tag,tag_link):
        sql="insert into tags (qID,tag,tag_link) values (?,?,?)"
        self.cursor.execute(sql,[qID,tag,tag_link])

    def closeDB(self):
        #关闭数据库
        self.con.commit()
        self.con.close()

    def spider(self,url):
        #爬虫函数
        try:
            resp=urllib.request.urlopen(url)
            html=resp.read().decode()
            root=etree.HTML(html)
            divs=root.xpath("//div[@class='quote']")
            for div in divs:
                quote=div.xpath(".//span[@class='text']")[0].text
                author=div.xpath(".//small")[0].text
                print(quote,author)
                author_link=div.xpath(".//span[position()=2]//a/@href")[
0]
                print(author_link)
                ID=self.insertQuote(quote,author,author_link)
```

```
                print(quote,author,author_link)
                links= div.xpath(".//div[@class='tags']//a")
                for link in links:
                    self.insertTag(ID,link.text,link.attrib.get("href"))
        except Exception as err:
            print(err)

    def show(self):
        #显示函数
        self.cursor.execute("select ID,quote,author,author_link from
quotes")
        quotes=self.cursor.fetchall()
        no=0
        for quote in quotes:
            no=no+1
            print("No",no)
            ID=quote[0]
            print(quote[1])
            print(quote[2])
            print(quote[3])
            self.cursor.execute("select qID,tag,tag_link from tags where
qID="+str(ID))
            tags=self.cursor.fetchall()
            for tag in tags:
                print(tag[1]," --- ",tag[2])
            print()
        print("Total ",len(quotes))

#主程序
spider=MySpider()
spider.openDB()
spider.spider("http://127.0.0.1:5000")
#spider.spider("http://www.rj-jwb.com.cn/spiderbook/chapter2/")
spider.show()
spider.closeDB()
```

4. 下面是一段 HTML 编码：

```
<body>
<bookstore>
<book id="b1">
  <title lang="english">Harry Potter</title>
  <price>23.99</price>
```

```
    </book>
    <book id="b2">
      <title lang="chinese">学习 XML</title>
      <price>39.95</price>
    </book>
    <book id="b3">
      <title lang="english">Learning Python</title>
      <price>30.20</price>
    </book>
    </bookstore>
    </body></html>
```

试用 xpath 完成下面任务。

(1) 找出所有书的名称。

(2) 找出所有英文书的名称与价格。

(3) 找出价格在 30 元以上的所有书的名称。

答：

```
from lxml import etree
html='''
<body>
<bookstore>
<book id="b1">
  <title lang="english">Harry Potter</title>
  <price>23.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
  <price>39.95</price>
</book>
<book id="b3">
  <title lang="english">Learning Python</title>
  <price>30.20</price>
</book>
</bookstore>
</body></html>
'''
root=etree.HTML(html)
print("(1)")
titles=root.xpath('//book/title')
for title in titles:
    print(title.text)
print("(2)")
```

```
titles=root.xpath('//book/title[@lang="english"]')
for title in titles:
    price=title.getparent().xpath("./price")[0].text
    print(title.text,price)
print("(3)")
books=root.xpath('//book')
for book in books:
    price=float(book.xpath('./price')[0].text)
    if price>=30:
        title=book.xpath('./title')[0]
        print(title.text,price)
```

练习 4

1．什么是深度优先法和广度优先法，它们有什么特点？

答：省略

2.一般程序采用函数递归调用去访问各个网站，那么是深度优先还是广度优先方法？

答：省略

3．如何启动一个 Python 线程？为什么说爬虫程序一般都会使用多线程？

答：省略

4．设计两个网页 A.html 与 B.html，它们都包含相同结构的学生信息（姓名、性别、年龄）。A.html 文件与 B.html 文件如下。

（1）A.html：

```
<div>姓名：A1</div><div>性别：男</div><div>年龄：20</div>
…
<div>姓名：A10</div><div>性别：女</div><div>年龄：18</div>
```

（2）B.html：

```
<div>姓名：B1</div><div>性别：男</div><div>年龄：20</div>
…
<div>姓名：B10</div><div>性别：女</div><div>年龄：18</div>
```

设计一个包含两个线程的爬虫程序，一个线程爬取 A.html，另外一个线程爬取 B.html，爬取的学生信息都保存到列表 students 中，最后显示爬取的结果。

答：

(1) server.py

```
import flask

app=flask.Flask("web")
@app.route("/A")
def A():
    s='''
    <div>姓名：A1</div><div>性别：男</div><div>年龄：20</div>
    <div>姓名：A10</div><div>性别：女</div><div>年龄：18</div>
    '''
    return s

@app.route("/B")
def B():
    s='''
```

```
    <div>姓名: B1</div><div>性别: 男</div><div>年龄: 20</div>
    <div>姓名: B10</div><div>性别: 女</div><div>年龄: 18</div>
    '''
    return s

app.debug=True
app.run()
```

(2) spider.py

```
from bs4 import BeautifulSoup
import urllib.request
import threading

lock=threading.RLock()

def getValue(s):
    p=s.find(":")
    return s[p+1:]

def getAB(page):
    global students
    try:
        resp=urllib.request.urlopen("http://127.0.0.1:5000/"+page)
        html=resp.read().decode()
        soup=BeautifulSoup(html,"html.parser")
        divs=soup.find_all("div")
        i=0
        while i<len(divs):
            name=getValue(divs[i].text)
            sex=getValue(divs[i+1].text)
            age=getValue(divs[i+2].text)
            i=i+3
            lock.acquire()
            students.append([name,sex,age])
            lock.release()
    except Exception as err:
        print(err)

students=[]
TA=threading.Thread(target=getAB,args=["A"])
TB=threading.Thread(target=getAB,args=["B"])
TA.start()
```

```
TB.start()
TA.join()
TB.join()
print(students)
```

结果：

[[' B1', ' 男', ' 20'], [' B10', ' 女', ' 18'], [' A1', ' 男', ' 20'], [' A10', ' 女', ' 18']]

练习 5

1．简单说明 Scrapy 的入口地址的规定。

答：省略

2．比较使用 XPath 和 Beautiful Soup 解析 HTML 代码的特点与区别。

答：省略

3．在 Scrapy 中如何把爬取的数据写入数据库？items.py 与 pipelines.py 文件有什么作用？

答：

items.py 定义数据字段，pipelines.py 获取数据进行存储。

4．Python 中的 yield 语句如何工作？为什么 Scrapy 爬到的数据使用 yield 语句返回，而不是使用 return 语句返回？

答：省略

5．Scrapy 是如何实现爬取多个网页的数据的？如何理解分布式爬取过程？

答：省略

6．有一个服务器程序，它根据请求页面的不同来展示 3 组学生信息，这些学生信息分别存储在 students1.txt、students2.txt、students3.txt 文件中，每个页面有相同的结构（学号 No、姓名 Name、性别 Gender、年龄 Age），参数 page=N 控制展示 studentsN.txt 的学生表格（N=1,2,3），程序如下：

```
    import flask
    app=flask.Flask(__name__)
    @app.route("/")
    def show():
        page = flask.request.args.get("page") if "page" in flask.request.
args else "1"
        maxpage=3
        page=int(page)
        st="<h3>学生信息表</h3>"
        st=st+"<table border='1' width='300'>"
        fobj=open("students"+str(page)+".txt","rt",encoding="utf-8")
        while True:
            #读取一行，去除行尾部“\n”换行符号
            s=fobj.readline().strip("\n")
            #如果读到文件尾部就退出
            if s=="":
```

```
                break
            #按逗号拆分
            s=s.split(",")
            st=st+"<tr>"
            #把各个数据组织在<td>...</td>中
            for i in range(len(s)):
                st=st+"<td>"+s[i]+"</td>"
            #完成一行
            st=st+"</tr>"
        fobj.close()
        st=st+"</table>"
        st=st+"<div>"
        if page>1:
            st = st + "<a href='/?page=" +str(page-1) + "'>【前一页】</a>"
        if page<maxpage:
            st=st+"<a href='/?page="+str(page+1)+"'>【下一页】</a>"
        st=st+"</div>"
        st=st+"<input type='hidden' name='page' value='"+str(page)+"'>"
        st=st+"<div>Page: "+str(page)+"/"+str(maxpage)+"</div>"
        return st

if __name__=="__main__":
    app.run()
```

执行服务器程序，学生信息如图 4-10-1 所示。

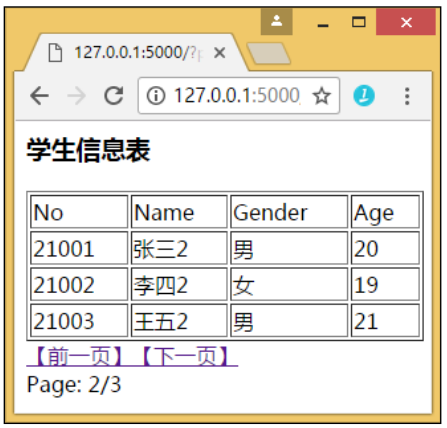


图 4-10-1　学生信息

使用 Scrapy 设计一个爬虫程序，爬取所有学生的信息并将其存储到数据库中。

答：

1、server.py

```
import flask
app=flask.Flask(__name__)
@app.route("/")
```

```
def show():
    page = flask.request.args.get("page") if "page" in
flask.request.args else "1"
    maxpage=3
    page=int(page)
    st="<h3>学生信息表</h3>"
    st=st+"<table border='1' width='300'>"
    fobj=open("students"+str(page)+".txt","rt",encoding="utf-8")
    while True:
        #读取一行，去除行尾部“\n”换行符号
        s=fobj.readline().strip("\n")
        #如果读到文件尾部就退出
        if s=="":
                break
        #按逗号拆分
        s=s.split(",")
        st=st+"<tr>"
        #把各个数据组织在<td>…</td>中
        for i in range(len(s)):
                st=st+"<td>"+s[i]+"</td>"
        #完成一行
        st=st+"</tr>"
    fobj.close()
    st=st+"</table>"
    st=st+"<div>"
    if page>1:
        st = st + "<a href='/?page=" +str(page-1) + "'>【前一页】</a>"
    if page<maxpage:
        st=st+"<a href='/?page="+str(page+1)+"'>【下一页】</a>"
    st=st+"</div>"
    st=st+"<input type='hidden' name='page' value='"+str(page)+"'>"
    st=st+"<div>Page: "+str(page)+"/"+str(maxpage)+"</div>"
    return st

if __name__=="__main__":
    app.debug=True
    app.run()
```

2、爬虫程序

scrapy startproject demo

(1) items.py

```
import scrapy

class StudentItem(scrapy.Item):
    # define the fields for your item here like:
    No = scrapy.Field()
    Name=scrapy.Field()
    Gender=scrapy.Field()
    Age=scrapy.Field()
```

(2) pipelines.py

```
import sqlite3

class DemoPipeline:
    def open_spider(self,spider):
        print("opened")
        try:
            self.con=sqlite3.connect("students.db")
            self.cursor=self.con.cursor()
            try:
                self.cursor.execute("drop table students")
            except:
                pass
            try:
                sql="""
                create table students
(
  No varchar(8) primary key,
  Name varchar(32),
  Gender varchar(8),
  Age int
)
                """
                self.cursor.execute(sql)
            except:
                self.cursor.execute("delete from students")
            self.opened=True
            self.count=0
        except Exception as err:
            print(err)
            self.opened=False

    def close_spider(self, spider):
        if self.opened:
            self.con.commit()
```

```
            self.con.close()
            self.opened=False
        print("closed")
        print("总共爬取",self.count,"条记录")


    def process_item(self, item, spider):
        if self.opened:
            try:
                sql="insert into students (No,Name,Gender,Age) values
(?,?,?,?)"
                self.cursor.execute(sql,[item["No"],item["Name"],item["G
ender"],item["Age"]])
                self.count=self.count+1
            except Exception as err:
                print(err)
        return item
```

(3) mySpider.py

```
import scrapy
from bs4 import BeautifulSoup
from items import StudentItem

class MySpider(scrapy.Spider):
    name = "mySpider"
    def start_requests(self):
        url ='http://127.0.0.1:5000'
        yield scrapy.Request(url=url,callback=self.parse)

    def parse(self, response):
        print(response.url)
        html=response.body.decode()
        soup=BeautifulSoup(html,"html.parser")
        trs=soup.find_all("tr")
        for tr in trs:
            tds=tr.find_all("td")
            item=StudentItem()
            item["No"]=tds[0].text
            item["Name"]=tds[1].text
            item["Gender"]=tds[2].text
            item["Age"]=int(tds[3].text)
```

```
            yield item
        links=soup.find("div").find_all("a")
        nextLink=""
        for link in links:
            if link.text=="【下一页】":
                nextLink=response.urljoin(link["href"])
                yield scrapy.Request(url=nextLink,callback=self.parse)
                break
```

(4) settings.py

```
BOT_NAME = 'demo'
SPIDER_MODULES = ['demo.spiders']
NEWSPIDER_MODULE = 'demo.spiders'
# Obey robots.txt rules
ROBOTSTXT_OBEY = True
# Configure item pipelines
# See https://docs.scrapy.org/en/latest/topics/item-pipeline.html
ITEM_PIPELINES = {
    'demo.pipelines.DemoPipeline': 300,
}
```

练习 6

1．说明 Selenium 的工作原理。为什么其能执行网页中的 JavaScript 程序?

答：省略

2．说明 Selenium 的等待方式。各种等待方式有什么特点？

答：省略

3．启动如下的服务器程序。

```
import flask
app=flask.Flask(__name__)
@app.route("/")
def index():
     s='''<body>
<span id="jMsg"></span>
</body>
<script>
    document.getElementById("jMsg").innerHTML="javascript 信息";
</script>
'''
     return s
app.run()
```

(1) 用 urllib 编写程序来获取网站的 HTML 代码。

(2) 用 Selenium 编写程序来获取网站的 HTML 代码。

通过这两种方法获取的 HTML 代码是否相同？为什么？

答：

```
from bs4 import BeautifulSoup
import urllib.request
from selenium import webdriver
from selenium.webdriver.chrome.options import Options

def testSelenium():
    try:
        chrome_options = Options()
        chrome_options.add_argument('--headless')
        chrome_options.add_argument('--disable-gpu')
        driver= webdriver.Chrome(chrome_options=chrome_options)
        driver.get("http://127.0.0.1:5000")
        html=driver.page_source
        soup=BeautifulSoup(html,"lxml")
        jMsg=soup.find("span",attrs={"id":"jMsg"})
        if jMsg:
```

```
            print("Selenium found ",jMsg.text)
        else:
            print("Selenium not found")
        driver.close()
    except Exception as err:
        print(err)

def testRequest():
    try:
        resp=urllib.request.urlopen("http://127.0.0.1:5000")
        html=resp.read().decode()
        soup=BeautifulSoup(html,"lxml")
        jMsg=soup.find("span",attrs={"id":"jMsg"})
        if jMsg:
            print("Request found ",jMsg.text)
        else:
            print("Request not found")
    except Exception as err:
        print(err)

testRequest()
testSelenium()
```

结果：

Request found

Selenium found　javascript 信息

4．启动如下的服务器程序。

```
import flask
app=flask.Flask(__name__)
@app.route("/")
def index():
     s='''<body>
<span id="jMsg"></span>
</body>
<script>
    function msg()
    {
       document.getElementById("jMsg").innerHTML="javascript 信息";
    }
    window.setTimeout(msg,1000);
</script>
'''
     return s
app.run()
```

试用 Selenium 编写爬虫程序等待…中出现的字符串，并爬取该字符串。

答：

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

def testSelenium():
    try:
        chrome_options = Options()
        chrome_options.add_argument('--headless')
        chrome_options.add_argument('--disable-gpu')
        driver= webdriver.Chrome(chrome_options=chrome_options)
        driver.get("http://127.0.0.1:5000")
        locator = (By.ID, "jMsg")
        WebDriverWait(driver,
10,0.5).until(EC.text_to_be_present_in_element(locator,"java"))
        msg=driver.find_element_by_id("jMsg").text
        print("Selenium found ",msg)
        driver.close()
    except Exception as err:
        print(err)

testSelenium()
```

5．启动如下的服务器程序。

```
import flask
app=flask.Flask(__name__)
@app.route("/")
def index():
     s='''<body>
<form name='frm' action='/handin' method='get'>
姓名<input type='text' name='name'><br>
性别<input type='radio' name='gender' value='男'>男<input type='radio'
name='gender' value='女'>女<br>
<input type='submit' value='提交' >
</form>
'''
     return s
```

```
    @app.route("/handin")
    def handin():
         name=flask.request.values.get("name")
         gender = flask.request.values.get("gender")
         return "<div id='name'>"+name+"</div><div id='gender'>
"+gender+"</div>"

    app.run()
```

试用 Selenium 编写程序：

（1）填写姓名和性别为“张三”与“女”，并提交；

（2）爬取提交后的姓名与性别。

答：

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import time

def testSelenium():
    try:
        chrome_options = Options()
        chrome_options.add_argument('--headless')
        chrome_options.add_argument('--disable-gpu')
        driver= webdriver.Chrome(chrome_options=chrome_options)
        #driver= webdriver.Chrome()
        driver.get("http://127.0.0.1:5000")
        nameElem=driver.find_element_by_css_selector("input[name='name'
]")
        genderElem=driver.find_element_by_css_selector("input[name='gen
der'][value='女']")
        submitElem=driver.find_element_by_css_selector("input[type='sub
mit']")
        nameElem.send_keys("张三")
        genderElem.click()
        submitElem.click()
        time.sleep(1)
        name=driver.find_element_by_id("name").text
        gender=driver.find_element_by_id("gender").text
        print(name,gender)
    except Exception as err:
        print(err)

testSelenium()
```

