# 第四阶段 Python数据分析

# Python 爬虫2

## 网校相关视频

Python爬虫开发

第一章网络爬虫基础

## 明确本次课知识点，明确重点难点

### 【知识点目标】

* 使用代理服务器
* Python异常处理
* 利用正则表达式爬取数据

### 【重点】

* Python异常处理
* 利用正则表达式爬取数据

### 【难点】

* Python异常处理
* 利用正则表达式爬取数据

## 复习巩固作业讲解

## 本次课程任务讲解

### 【知识点1】Python爬虫突破封禁--使用代理服务器

由于一直用同一个IP爬取目标网站的数据，如果访问次数过多，目标网站服务器会禁止你的访问，所以需要经常更换自己IP。这时候就需要代理服务器了。

1. 代理服务器原理

匿名

防火墙

1. 爬虫为何要使用代理服务器

防BAN

稳定高速

1. 代理服务器的分类

透明代理

匿名代理

高匿代理：高度匿名

常用的免费代理网站：

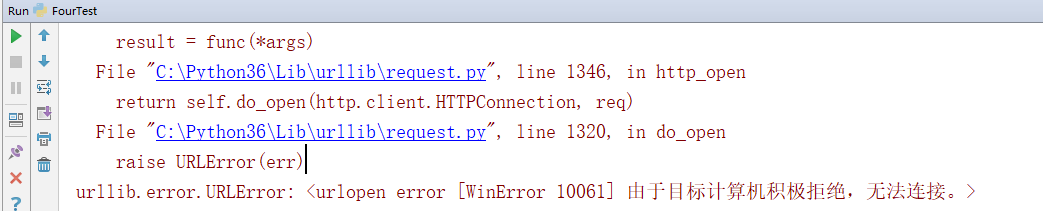
<https://www.kuaidaili.com>

<https://www.xicidaili.com/>

"118.190.95.35:9001","211.159.171.58:80"

1. 使用代理进行爬取的步骤
   1. 构建代理池
   2. 从代理池中随机选取代理服务器，并利用urllib.request.ProxyHandler方法生成代理处理器
   3. 根据urllib.request.build\_opener创建支持处理HTTP请求的opener对象
   4. 设置headers常用属性
   5. 通过opener对象的addheaders属性将headers常用属性赋值，为请求对象添加header
   6. 调用opener对象的open方法，根据请求建立连接，获取响应结果

### 【知识点2】Python异常处理





异常就是程序运行时发生错误的信号，python用异常对象（exception object）来表示异常情况。遇到错误后，会引发异常。如果异常对象并未被处理或捕捉，程序就会用所谓的“回溯”（Traceback， 一种错误信息）终止执行。但实际程序运行过程中，有时候我们并不希望异常导致程序运行中断，此时就可以用异常处理来对程序进行控制，即专门书写一段程序来对程序触发的异常进行捕捉，一旦捕捉成功则进入专门的处理分支，执行程序员设定的功能，使程序不会崩溃，这就是异常处理。

1. Python程序中的错误
   1. 程序错误的分类
      1. 语法错误

是指不遵循语言的语法结构引起的错误（程序无法正常编译/运行）。

在执行前必须解决

常见的 Python 语法错误有：

* 遗漏了某些必要的符号（冒号、逗号或括号）
* 关键字拼写错误
* 缩进不正确
* 空语句块（需要用 pass 语句）
  + 1. 逻辑错误

（也称：语义错误）：是指程序的执行结果与预期不符（程序可以正常运行，不会 Crash）

不容易发现

常见的 Python 逻辑错误有：

* 运算符优先级考虑不周
* 变量名使用不正确
* 语句块缩进层次不对
* 在布尔表达式中出错
  + 1. 运行时错误

是指程序可以运行，但是在运行过程中遇到错误，导致意外退出。在 Python 中，这种运行时错误被称为**异常**。

异常可以通过代码解决

常见的Python内置异常

*- BaseException # 所有异常的基类*

*- Exception # 常规异常的基类（万能异常）*

*- ArithmeticError # 各种算数错误引起的异常*

*- FloatingPointError # 浮点数操作错误*

*- OverflowError # 结果超出范围*

*- ZeroDivisionError # ０为除数异常*

*- AttributeError # 属性引用异常*

*- BufferError # 缓存错误*

*- EOFError # 读不到数据*

*- ImportError # import错误*

*- LookupError # 由索引和key值引起的异常*

*- IndexError # 索引错误*

*- KeyError # 字典key值错误*

*- MemortError # 内存溢出异常*

*- NameError # 本地和全局找不到变量名*

*- UnboundLocalError # 局部变量没有赋值*

*- OSError # system错误*

*- ConnectionError # 连接*

*- BrokenPipeError # 管道读写异常*

*- ConnectionAbortedError # 连接失败*

*- ConnectionRefusedError # 连接拒绝*

*- ConnectionResetError # 连接重置*

*- FileExistsError # 创建文件和文件夹错误*

*- FileNotFoundError # 文件未找到*

*- PermissionError # 权限*

*- ProcessLookupError # 进程不存在*

*- TimeoutError # 超时*

*- SyntaxError # 语法错误*

*- IndentationError # 缩进错误*

*- SystemError # 解释器中断*

*- TypeError # 类型错误*

*- ValueError # 赋值错误*

*- UnicodeError # 和unicode相关的错误*

*- UnicodeEncodeError # unicode编码错误*

*- UnicodeDecodeError # unicode解码错误*

*- UnicodeTranslateError # unicode转换错误*

*- Warning # 警告的基类*

*- DeprecationWarning # 操作不赞成警告*

*- PendingDeprecationWarning # 表明此操作将来会被弃用*

*- UserWarning # 用于用户生成警告*

*- SyntaxWarning # 语法可疑警告*

*- RuntimeWarning # 运行警告*

*- FutureWarning # 将会改变警告*

*- ImportWarning # 导入警告*

*- UnicodeWarning # unicode相关警告*

*- BytesWarning # 字节相关警告*

*- ResourceWarning # 资源使用情况警告*

*参考：*

*<https://www.runoob.com/python/python-exceptions.html>*

*<https://docs.python.org/zh-cn/3.7/library/exceptions.html>*

1. 异常处理
   1. 作用：如果你不想在异常发生时结束你的程序，可以使用try/except语句捕捉异常，当异常捕捉到时程序可不终止。
   2. 关键字：try检测语句块中的错误，让except语句捕获异常信息并处理。
   3. 语法结构

try :

<语句> # 运行的代码

except <异常类> [as 变量名]:

<语句> # 如果发生指定的异常，则执行相应的语句

else:

<语句> # 如果没有发生指定的异常，则执行相应的语句

finally:

<语句> # 不管是否发生指定的异常，最后都会执行的语句

1. 爬取数据时的异常处理

在网页的访问过程中经常会碰到一些异常情况的发生，编写代码时如若不做处理，异常后的代码将执行不到，为不影响整个项目的执行，需要有异常处理机制。

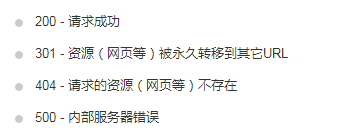
* 1. 爬数据时常见的异常
     1. URLError

URLError可能产生的常见原因包括：网络无连接，即本机无法上网；连接不到指定的服务器；服务器不存在等。

* + 1. HTTPError

HTTPError是URLError的子类，在利用urlopen方法发出一个请求时，服务器上都会对应一个应答对象response，其中它包含一个数字”状态码”，它表示HTTP协议所返回的响应的状态。

常见的HTTP状态码



1\*\*(信息类)：表示接收到请求并且继续处理

100——客户必须继续发出请求

101——客户要求服务器根据请求转换HTTP协议版本

2\*\*(响应成功)：表示动作被成功接收、理解和接受

200——表明该请求被成功地完成，所请求的资源发送回客户端

201——提示知道新文件的URL

202——接受和处理、但处理未完成

203——返回信息不确定或不完整

204——请求收到，但返回信息为空

205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件

206——服务器已经完成了部分用户的GET请求

3\*\*(重定向类)：为了完成指定的动作，必须接受进一步处理

300——请求的资源可在多处得到

301——本网页被永久性转移到另一个URL

302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始URL地址，重定向，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。

303——建议客户访问其他URL或访问方式

304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用

305——请求的资源必须从服务器指定的地址得到

306——前一版本HTTP中使用的代码，现行版本中不再使用

307——申明请求的资源临时性删除

4\*\*(客户端错误类)：请求包含错误语法或不能正确执行

400——客户端请求有语法错误，不能被服务器所理解

401——请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

403——请求被服务器拒绝

404——一个404错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的URL

405——用户在Request-Line字段定义的方法不允许

406——根据用户发送的Accept拖，请求资源不可访问

407——类似401，用户必须首先在代理服务器上得到授权

408——客户端没有在用户指定的饿时间内完成请求

409——对当前资源状态，请求不能完成

410——服务器上不再有此资源且无进一步的参考地址

411——服务器拒绝用户定义的Content-Length属性请求

412——一个或多个请求头字段在当前请求中错误

413——请求的资源大于服务器允许的大小

414——请求的资源URL长于服务器允许的长度

415——请求资源不支持请求项目格式

416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段

417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。

5\*\*(服务端错误类)：服务器不能正确执行一个正确的请求

500 - 服务器遇到错误，无法完成请求

501 - 未实现

502 - 网关错误

503 - 由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常

参考：<https://www.runoob.com/http/http-status-codes.html>

以上的状态码中以3开头的代号可以浏览器被处理，100-299范围的号码指示成功，而400-599表示错误号码，将可以被程序员看出。使用code属性可以获取状态码

### 【知识点3】使用正则表达式爬取数据

1. 正则表达式的概念

正则表达式是一个特殊的字符序列，它能帮助你方便的检查一个字符串是否与某种模式匹配。

1. Python中的Re模块

提供了对正则表达式的支持

返回pattern对象：

***re.compile(string[,flag])***

匹配函数：

***re.match(pattern, string[, flags]) re.search(pattern, string[, flags])***

***re.split(pattern, string[, maxsplit])***

***re.findall(pattern, string[, flags])***

***re.finditer(pattern, string[, flags])***

***re.sub(pattern, repl, string[, count])***

***re.subn(pattern, repl, string[, count])***

1. re.compile

***compile(pattern[,flags] )***

根据包含正则表达式的字符串创建模式对象

flag参数：匹配模式，取值可以使用按位或运算符’|’表示同时生效，比如re.I | re.M。可选值有：

（1）re.I(全拼：IGNORECASE): 忽略大小写

（2）re.M(全拼：MULTILINE): 多行模式，改变'^'和'$'的行为

（3）re.S(全拼：DOTALL): 点任意匹配模式，改变'.'的行为

（4）re.L(全拼：LOCALE): 使预定字符类 \w \W \b \B \s \S 取决于当前区域设定

（5）re.U(全拼：UNICODE): 使预定字符类 \w \W \b \B \s \S \d \D 取决于unicode定义的字符属性

（6）re.X(全拼：VERBOSE): 详细模式。这个模式下正则表达式可以是多行，忽略空白字符，并可以加入注释。

1. 匹配
   1. ***re.match()：***从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话返回None
2. 正则表达式语法：

正则表达式通常通过特殊的语法来表示。方法如下：

* 1. ^表示字符串的开始，$表示字符串的末尾，经常省略
  2. 字母和数字表示他们自身。一个正则表达式模式中的字母和数字匹配同样的字符串。
  3. 其它模式特殊符号描述如下：







1. 使用正则表达式：
   1. **re*.search(pattern, string[, flags])***

re.search函数会在字符串内查找模式匹配,只要找到第一个匹配然后返回，如果字符串没有匹配，则返回None。

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。

match和search一旦匹配成功，就是一个match object对象，而match object对象有以下方法：

group() 返回被 RE 匹配的字符串

start() 返回匹配开始的位置

end() 返回匹配结束的位置

span() 返回一个元组包含匹配 (开始,结束) 的位置

group() 返回re整体匹配的字符串，可以一次输入多个组号，对应组号匹配的字符串。

* 1. ***re.findall(pattern, string[, flags])***

遍历匹配，可以获取字符串中所有匹配的字符串，返回一个列表。(在网页爬取中最常用)

* 1. ***re.finditer(pattern, string[, flags])***

搜索string，返回一个顺序访问每一个匹配结果（Match对象）的迭代器。找到 RE 匹配的所有子串，并把它们作为一个迭代器返回。

* 1. ***re.split(pattern, string[, maxsplit])***

maxsplit用于指定最大分割次数，不指定将全部分割。

按照能够匹配的子串将string分割后返回列表。可以使用re.split来分割字符串，如：re.split(r'\s+', text)；将字符串按空格分割成一个单词列表。

* 1. ***re.sub(pattern, repl, string, count)***

使用re替换string中每一个匹配的子串后返回替换后的字符串。

* 1. ***re.subn(pattern, repl, string, count=0, flags=0)***

返回替换次数

## 自主学习作业讲解

无

## 课程总结

#### 使用代理服务器

#### Python中的异常处理

#### 使用正则表达式处理爬取的网页数据

## 下次自主学习任务布置

编写代码实现下列需求

爬取51job上的职位信息，并写入到文本和excel文件中

