# 课程介绍

* 登录过程原理（了解）
* Cookie（掌握）
* Handler处理器和自定义Opener（了解）
* http.cookieJar模块（掌握）
* 模拟登录（掌握）
* 第三方库Requests的使用（掌握）

1. 登陆过程原理
   1. 引入原因

有些页面需要登陆才能获取到。比如，知乎的部分内容、新浪微博上非公众号的微博详情等。

* 1. 原理
     1. 注册过程

注册时，靠谱的服务器端不会明文保存用户的密码，而是保存用户密码的不可逆hash值（有时候还会将用户密码和一个有一定随机性的字符串进行某种拼接，称为加盐，然后再做hash）。

* + 1. 登陆过程

用户输入用户名、密码后，密码将按照指定规则计算得到不可逆的hash值，由服务器端保存。将来登录的时候，客户端会接受用户名、密码，按照服务器指定规则计算出一个hash值，通过POST发送给服务器，服务器端进行验证后，如果验证通过，就会发送一个带token的响应，这个token就是登陆后Cookie的最重要组成部分；如果未验证通过，则登录失败，提示用户名或密码错误。

**注意：避免使用Get方式提交表单，因为有可能会导致安全问题。 比如说在登陆表单中用Get方式，用户输入的用户名和密码将在地址栏中暴露无遗。登陆必然涉及用户名和密码等敏感信息，所以一定要使用POST请求。**

* + 1. HHHdCookie 和 Session：

服务器和客户端的交互仅限于请求/响应过程，结束之后便断开，在下一次请求时，服务器会认为新的客户端。

为了维护他们之间的链接，让服务器知道这是前一个用户发送的请求，必须在一个地方保存客户端的信息。

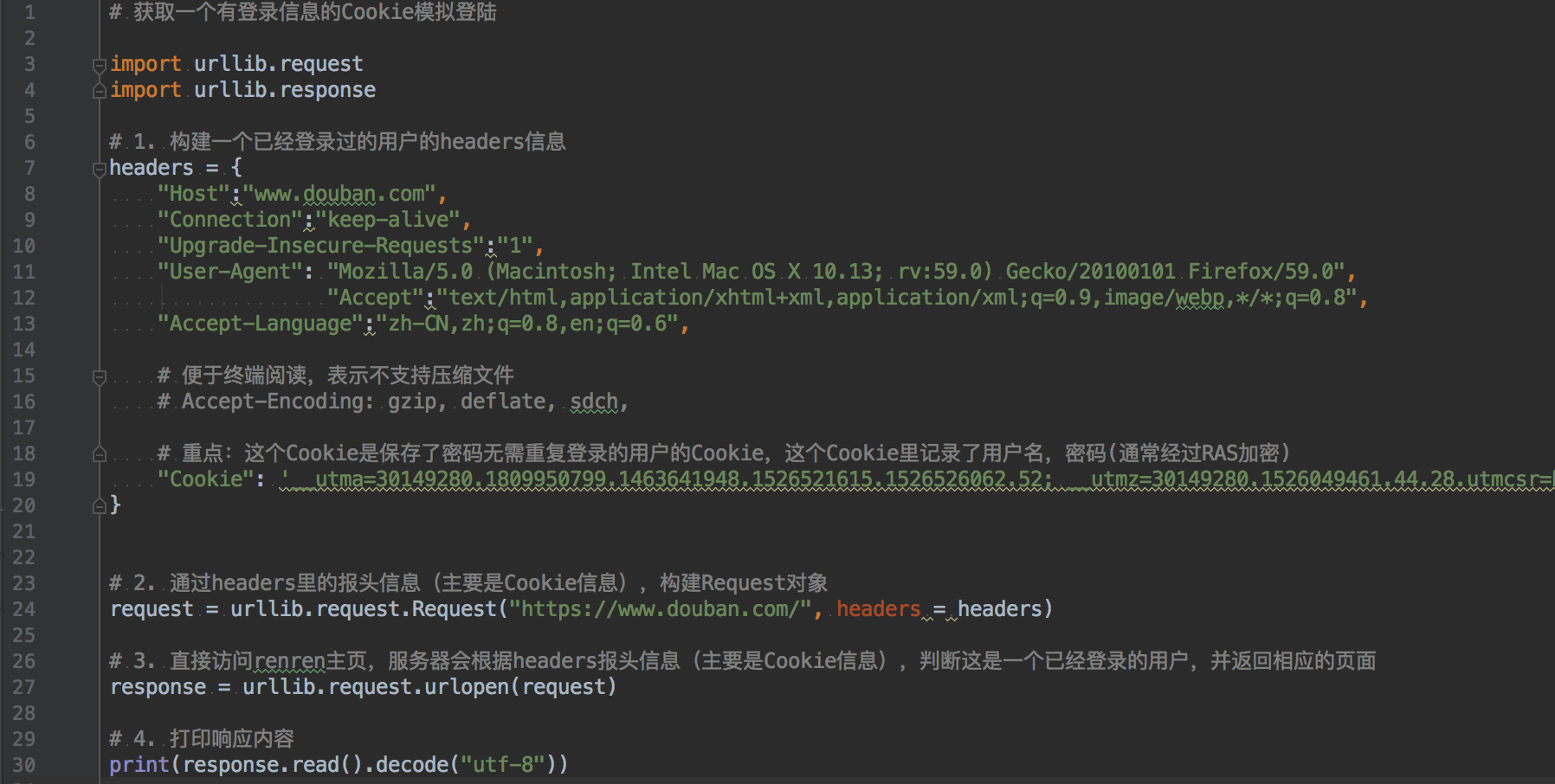
**Cookie**：通过在 客户端 记录的信息用来证明用户的身份。

**Session**：通过在 服务器端 记录的信息用来验证用户的身份。

* 1. 使用

Cookies在爬虫方面最典型的应用是判定注册用户是否已经登录网站，用户可能会得到提示，是否在下一次进入此网站时保留用户信息以便简化登录手续。

**案例1 使用Cookie模拟登陆网站**



但是这样做太过复杂，我们先需要在浏览器登录账户，并且通过分析请求才能获取这个Cookie，那有么有更简单方便的方法呢？人工操作太多，爬虫的自动化程度应该更高。

1. 自定义Opener（请求发送器）和Handler（处理器）

Opener是urllib.OpenerDirector的子类，之前使用的urlopen就是一种opener。

* 1. 自定义Opener
     1. 为什么要用Opener

基本的urlopen()方法不支持代理、cookie等其他的HTTP/HTTPS高级功能。

* + 1. 使用Opener的步骤

1. 使用相关的 Handler处理器来创建特定功能的处理器对象；
2. 然后通过 urllib.build\_opener()方法使用这些处理器对象，创建自定义opener对象；
3. 使用自定义的opener对象，调用open()方法发送请求。
   * 1. 案例. 简单自定义opener()

用**urllib.request.build\_opener(…)**函数创建一个**Opener**。



这种方式发送请求得到的结果，和使用urllib.urlopen()发送HTTP/HTTPS请求得到的结果是一样的。

如果在 HTTPHandler()增加 debuglevel=1参数，还会将 Debug Log 打开，这样程序在执行的时候，会把收包和发包的报头在屏幕上自动打印出来，方便调试，有时可以省去抓包的工作。

那么HTTPHandler处理器是什么？接下来章节就会讲解。

* 1. Handler处理器

Handler处理器来创建具有特定功能的处理器对象，常用的有如下几种：

* + 1. ProxyHandler处理器（代理设置）

使用代理IP，这是爬虫/反爬虫的第二大招，通常也是最好用的。

很多网站会检测某一段时间某个IP的访问次数(通过流量统计，系统日志等)，如果访问次数多的不像正常人，它会禁止这个IP的访问。

所以我们可以设置一些代理服务器，每隔一段时间换一个代理，就算IP被禁止，依然可以换个IP继续爬取。

urllib中通过ProxyHandler来设置使用代理服务器，下面代码说明如何使用自定义opener来使用代理：

**案例3 使用ProxyHandler设置代理**



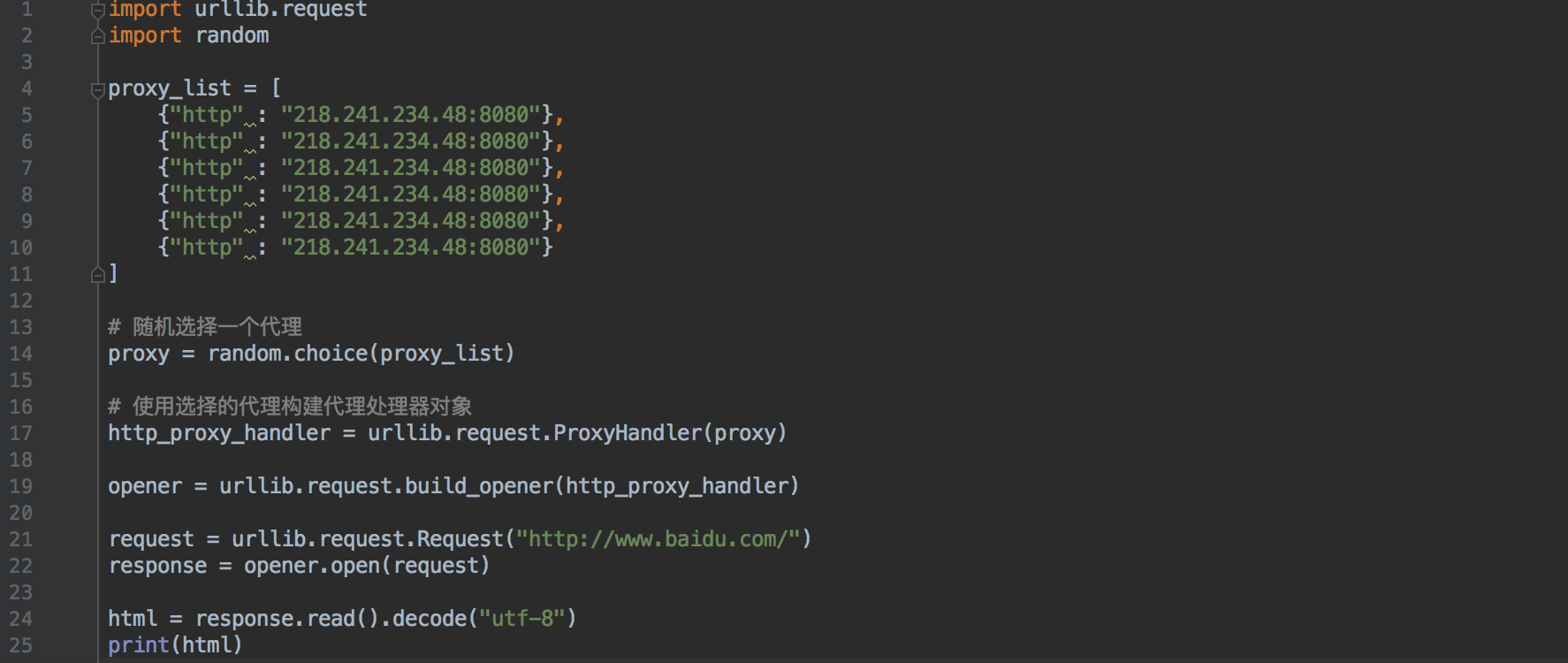
免费的开放代理获取基本没有成本，我们可以在一些代理网站上收集这些免费代理，测试后如果可以用，就把它收集起来用在爬虫上面。

免费短期代理网站举例：

* [西刺免费代理IP](http://www.xicidaili.com/)
* [快代理免费代理](http://www.kuaidaili.com/free/inha/)
* [Proxy360代理](http://www.proxy360.cn/default.aspx)
* [全网代理IP](http://www.goubanjia.com/free/index.shtml)

如果代理IP足够多，可以随机选择一个代理去访问网站。

**案例4 随机选取代理IP访问网站**



这类免费开放代理一般会有很多人都在使用，而且代理有寿命短，速度慢，匿名度不高，HTTP/HTTPS支持不稳定等缺点。

所以，专业爬虫工程师或爬虫公司会使用高品质的私密代理，这些代理通常需要找专门的代理供应商购买，再通过用户名/密码授权使用。而免费开放代理不需要用户名/密码授权使用。

* + 1. ProxyBasicAuthHandler（代理授权验证）

用来验证代理授权的用户名和密码。

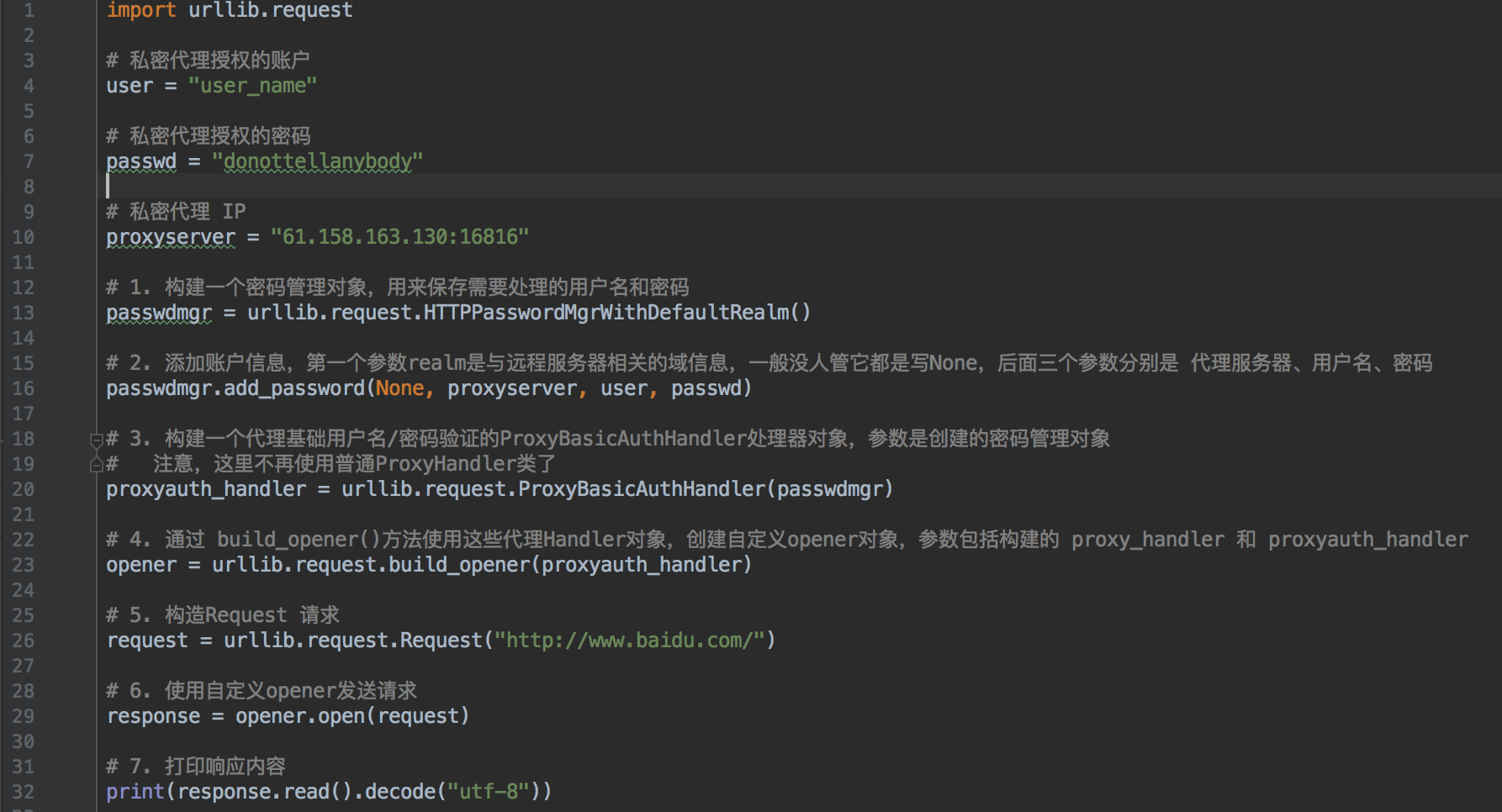
如果我们使用之前的代码来使用私密代理，会报 HTTP 407 错误，表示代理没有通过身份验证：

urllib2.HTTPError: HTTP Error 407: Proxy Authentication Required

所以我们需要改写代码，通过：

* + - 1. ProxyBasicAuthHandler（代理授权验证）HTTPPasswordMgrWithDefaultRealm()：来保存私密代理的用户密码
      2. ProxyBasicAuthHandler()：来处理代理的身份验证。

**案例5 通过授权验证代理访问网站**



1. Cookie

Cookie 是指某些网站服务器为了辨别用户身份和进行Session跟踪，而储存在用户浏览器上的文本文件，Cookie可以保持登录信息到用户下次与服务器的会话。

* 1. Cookie包含内容的解析

HTTP是无状态的面向连接的协议, 为了保持连接状态, 引入了Cookie机制 Cookie是http消息头中的一种属性，包括：

Cookie名字（Name）

Cookie的值（Value）

Cookie的过期时间（Expires/Max-Age）

Cookie作用路径（Path）

Cookie所在域名（Domain），

使用Cookie进行安全连接（Secure）。

前两个参数是Cookie应用的必要条件，另外，还包括Cookie大小（Size，不同浏览器对Cookie个数及大小限制是有差异的）。

Cookie由变量名和值组成，根据 Netscape公司的规定，Cookie格式如下：

Set－Cookie: NAME=VALUE；Expires=DATE；Path=PATH；Domain=DOMAIN\_NAME；SECURE

* 1. 使用cookielib库 和 HTTPCookieProcessor处理器

在Python处理Cookie，一般是通过cookielib模块和 urllib模块的HTTPCookieProcessor处理器类一起使用。

* + - 1. cookielib模块：主要作用是提供用于存储cookie的对象
      2. HTTPCookieProcessor处理器：主要作用是处理这些cookie对象，并构建handler对象。
    1. cookielib 库

该模块主要的对象有CookieJar、FileCookieJar、MozillaCookieJar、LWPCookieJar。

* + - 1. CookieJar：管理HTTP cookie值、存储HTTP请求生成的cookie、向传出的HTTP请求添加cookie的对象。整个cookie都存储在内存中，对CookieJar实例进行垃圾回收后cookie也将丢失。
      2. FileCookieJar (filename,delayload=None,policy=None)：从CookieJar派生而来，用来创建FileCookieJar实例，检索cookie信息并将cookie存储到文件中。filename是存储cookie的文件名。delayload为True时支持延迟访问访问文件，即只有在需要时才读取文件或在文件中存储数据。
      3. MozillaCookieJar (filename,delayload=None,policy=None)：从FileCookieJar派生而来，创建与Mozilla浏览器 cookies.txt兼容的FileCookieJar实例。
      4. LWPCookieJar (filename,delayload=None,policy=None)：从FileCookieJar派生而来，创建与libwww-perl标准的 Set-Cookie3 文件格式兼容的FileCookieJar实例。

**其实大多数情况下，我们只用CookieJar()，如果需要和本地文件交互，就用 MozillaCookjar() 或 LWPCookieJar()**

我们来做几个案例：

* + 1. 案例 获取Cookie，并保存到CookieJar()对象中



我们使用以上方法将Cookie保存到cookiejar对象中，然后打印出了cookie中的值，也就是访问百度首页的Cookie值。

运行结果如下：

BAIDUID=E55EF37C43C6BF3EC5C7DBE6929F84C5:FG=1;BIDUPSID=E55EF37C43C6BF3EC5C7DBE6929F84C5;H\_PS\_PSSID=1468\_21079\_26350\_26433\_20930;PSTM=1526545819;BDSVRTM=0;BD\_HOME=0

* + 1. 案例 访问网站获得cookie，并把获得的cookie保存在cookie文件中

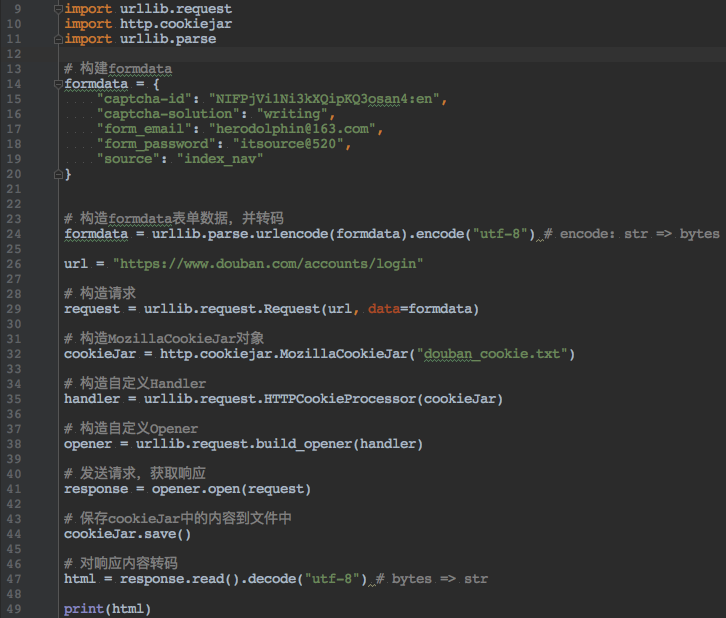


* + 1. 案例 从文件中获取cookies，做为请求的一部分去访问



我们使用以上方法将Cookie保存到cookiejar对象中，然后打印出了cookie中的值，也就是访问百度首页的Cookie值。

* + 1. 案例 利用cookielib登录douban



我们使用以上方法将Cookie保存到cookiejar对象中，然后打印出了cookie中的值，登录豆瓣后的Cookie值。

1. Requests: 让 HTTP 服务人类

虽然Python的标准库中 urllib 模块已经包含了平常我们使用的大多数功能，但是它的 API 使用起来让人感觉不太好，而 Requests 自称 “HTTP for Humans”，说明使用更简洁方便。

* 1. Requests介绍

Requests 继承了urllib的所有特性。Requests支持HTTP连接保持和连接池，支持使用cookie保持会话，支持文件上传，支持自动确定响应内容的编码，支持国际化的 URL 和 POST 数据自动编码。

Requests的文档非常完备，中文文档也相当不错。Requests能完全满足当前网络的需求，支持Python 2.6-3.7，而且能在PyPy下完美运行。

开源地址：<https://github.com/kennethreitz/requests>

中文文档 API： <http://docs.python-requests.org/zh_CN/latest/index.html>

* 1. 安装方式

推荐使用pip安装安装：

$ pip install requests

* 1. 基本GET请求（headers参数 和 parmas参数）
     1. 最基本的GET请求可以直接用get方法

response = requests.get("http://www.baidu.com/")

# 也可以这么写

# response = requests.request("get", "http://www.baidu.com/")

* + 1. 添加 headers 和 查询参数

如果想添加 headers，可以传入headers参数来增加请求头中的headers信息。如果要将参数放在url中传递，可以利用 params 参数。



* + - 1. 使用response.text 时，Requests 会基于 HTTP 响应的文本编码自动解码响应内容，大多数 Unicode 字符集都能被无缝地解码。
      2. 使用response.content 时，返回的是服务器响应数据的原始二进制字节流，可以用来保存图片等二进制文件。
  1. 基本POST请求（data参数）
     1. 最基本的GET请求可以直接用post方法

response = requests.post("http://www.baidu.com/", data = data)

* + 1. 传入data数据

对于 POST 请求来说，我们一般需要为它增加一些参数。那么最基本的传参方法可以利用 data 这个参数。



# 如果是json文件可以直接显示

print(response.json())

* 1. 代理（proxies参数）

如果需要使用代理，你可以通过为任意请求方法提供 proxies 参数来配置单个请求：



也可以通过本地环境变量 HTTP\_PROXY 和 HTTPS\_PROXY 来配置代理：

export HTTP\_PROXY="http://118.31.220.3:8080"

export HTTPS\_PROXY="https://118.31.220.3:8080"

* 1. 私密代理验证

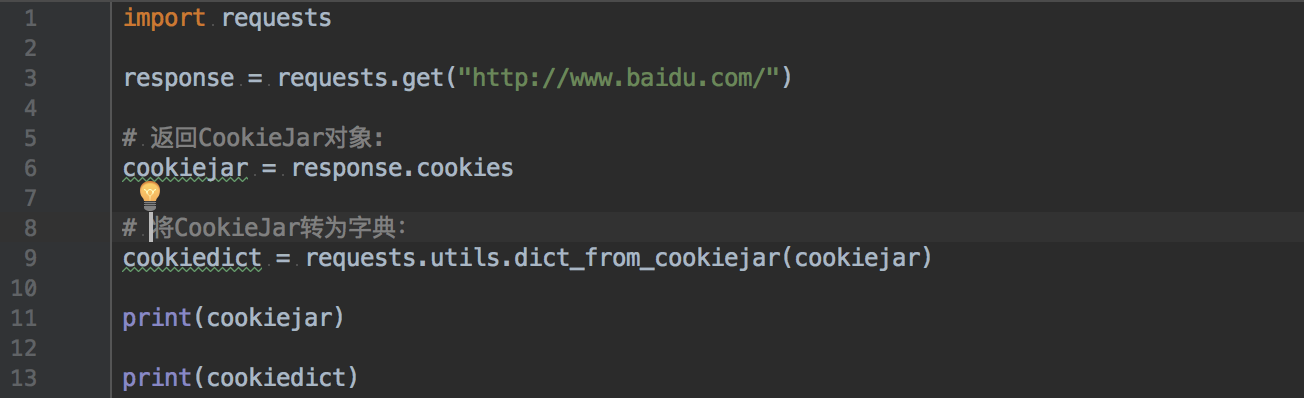
通过urllib使用私密代理比较复杂，而通过requests只需要一步：

* + 1. 私密代理



* 1. Cookies 和 Session
     1. Cookies

如果一个响应中包含了cookie，那么我们可以利用 cookies参数拿到：



* + 1. Session

在 requests 里，session对象是一个非常常用的对象，这个对象代表一次用户会话：从客户端浏览器连接服务器开始，到客户端浏览器与服务器断开。

会话能让我们在跨请求时候保持某些参数，比如在同一个 Session 实例发出的所有请求之间保持 cookie 。

* + 1. 案例 通过用户名、密码实现douban登录，并保存Cookie文件



1. 课程总结
   1. 重点
      * 1. 登录原理和过程
        2. Cookie和Session
        3. 利用cookie模拟登录
        4. 使用用户名、密码模拟登录
        5. 使用代理访问网站
        6. Requests库的使用
   2. 难点
      * 1. 使用用户名、密码模拟登录时反爬虫机制的应对
        2. cookie的准确获取
   3. 待抄写理论
      * 1. Opener和Handler使用的流程
        2. Requests库的常用函数和类的语法
2. 课后练习
   1. 必做题
      * 1. 实现上课代码，尽量领会后独立实现，包括自己组织语言做简要的注释
        2. 使用Cookie模拟登录自己感兴趣的网站，并证明完成了登录
        3. 使用用户名、密码模拟登录自己感兴趣的网站，通用需证明自己完成了登录
   2. 选做题
      * 1. 模拟登录12306网站。
3. 面试题
   1. Cookie的使用
4. 每日一练

复习HTML相关内容，重点是各种标签的含义和使用