Python网络爬虫与信息提取

# 爬虫的基本概念

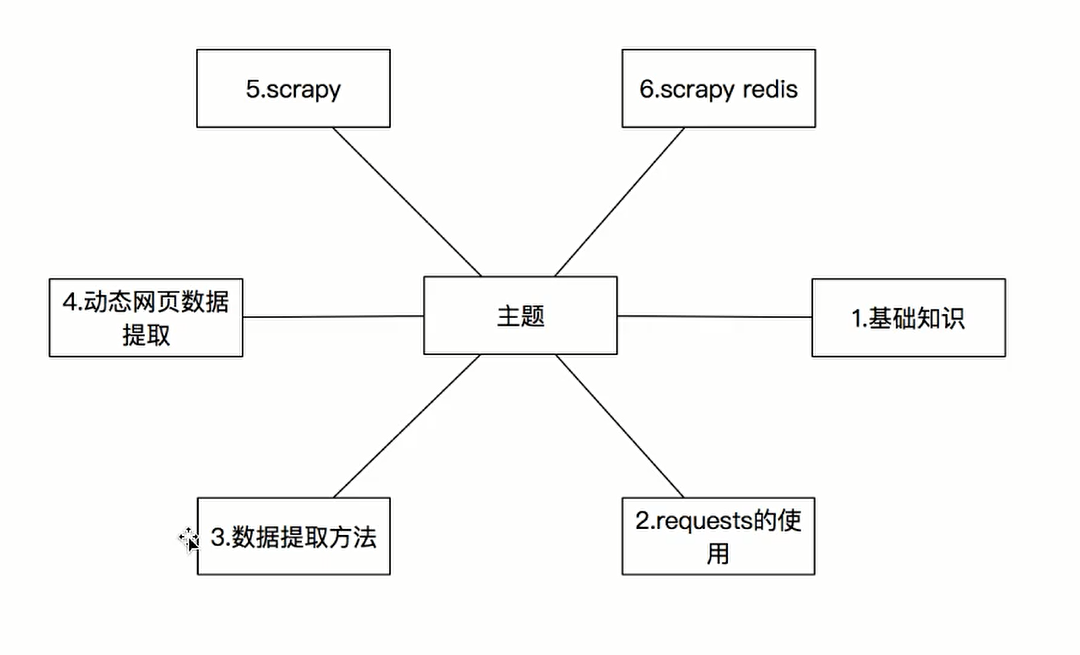
## 爬虫的概念

* 爬虫是模拟浏览器发送请求，获取响应

## 爬虫的流程

* url 🡪 发生请求，获取响应 🡪 提取数据 🡪 保存
* 发生请求，获取响应 🡪 提取url

## 主要内容



## 爬虫的应用场景



## 爬虫的定义

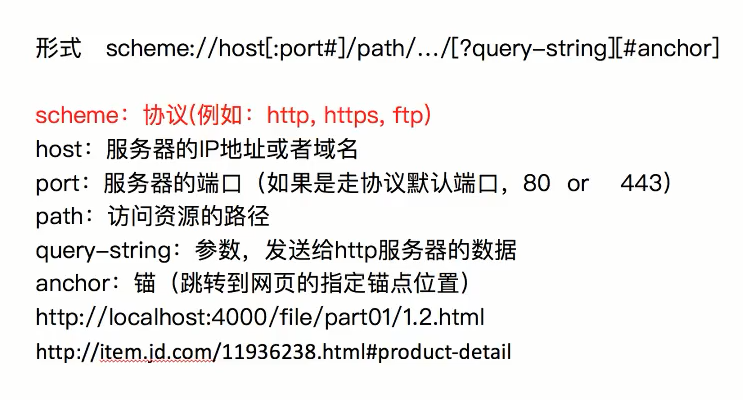
* 网络蜘蛛（又被称作网页蜘蛛，网络机器人）就是模拟客户端发送网络请求，接受请求响应，一种按照一定规则，自动的抓取互联网信息的程序。
* 只要浏览器能做的事，原则上，爬虫都可以做。

## 爬虫的更多用途

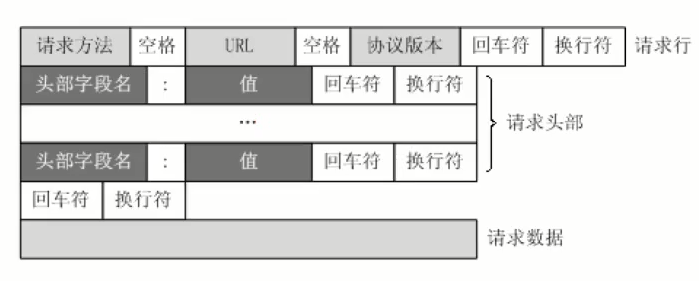
* 12306抢票
* 网站投票
* 短信轰炸

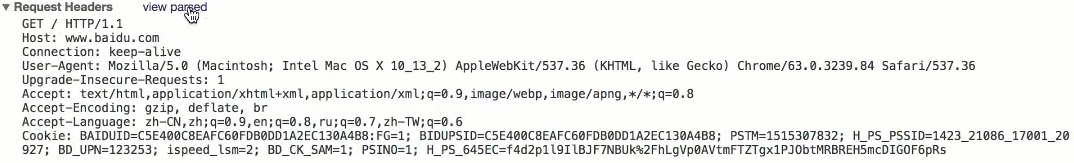
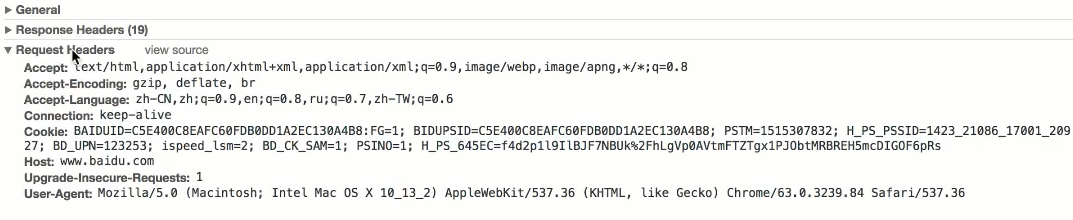
## url格式和http请求格式

* url的格式



* HTTP的请求形式





* 常见请求头



## 常用请求方法

* GET
* POST

问题：GET与POST方法有什么区别？

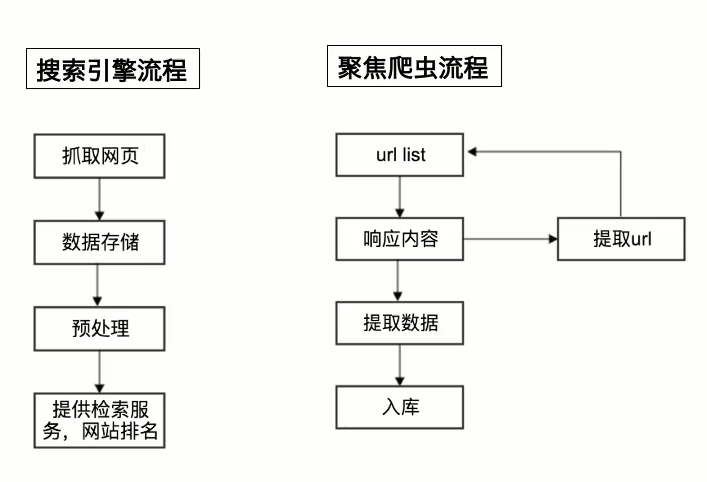
## 响应状态码（status code）

* 200：成功
* 302：临时转移至新的url
* 307：临时转移至新的url
* 404：not found
* 500：服务器内部错误

## 爬虫的分类

* 通用爬虫：通常指搜索引擎的爬虫
* 聚焦爬虫：针对特定网站的爬虫

## 通用爬虫和聚焦爬虫工作流程

通用搜索引擎的局限性

* 返回网页里90%的内容没有用
* 图片、音频、视频多媒体的内容通用搜索引擎无能为力
* 不同用户搜索的目的不全相同，但是返 容可能相同

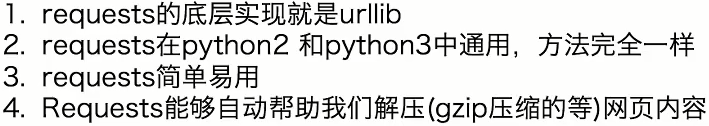
## Robots协议

网站通过Robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取。

例如：http：//www.taobao.com/robots.txt

# Requests使用入门

* 问题：为什么要学习requests，而不是urllib？



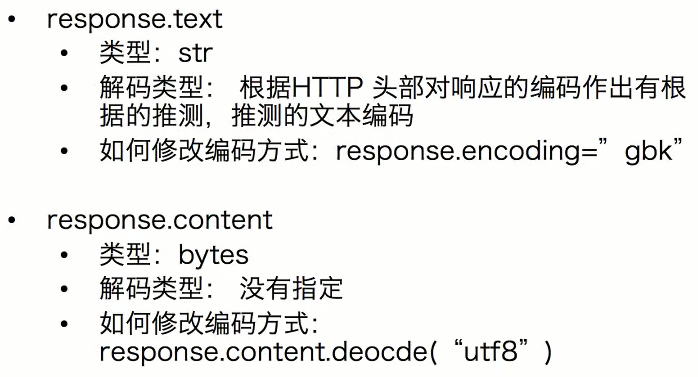
## Requests的作用

* 作用：发送网络请求，返回响应数据
* 中文文档 API：<http://doc.python-requests.org/zh_CN/lastest/index.html>

## Requests中的解码方法

* Response.content.decode()
* Response.content.decode(“gbk”)
* Response.text
* Response.encoding = “utf-8”

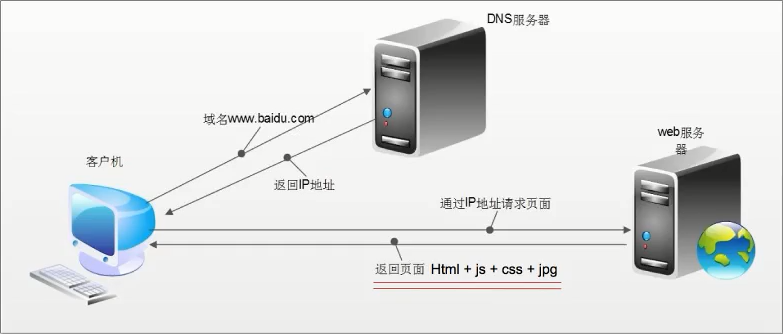
## Response.text与response.content的区别



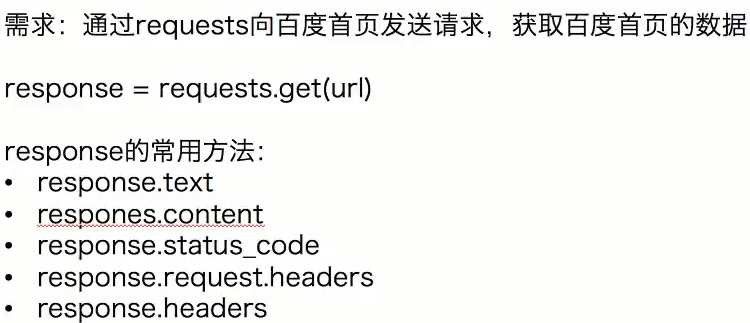
## Requests保存图片



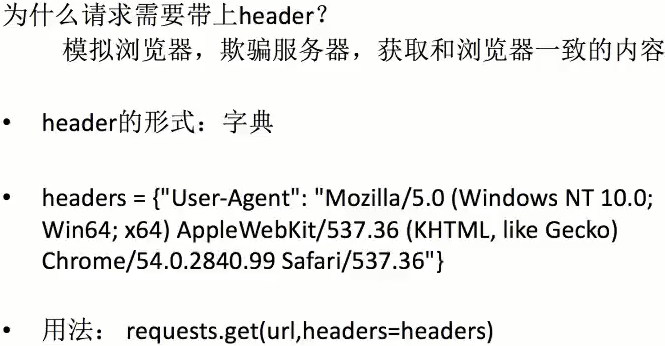
## 浏览器发送HTTP请求的过程



## 发送简单请求



## 发送带header请求



## 发送带参数的请求



## 发送Post请求

* 登录注册（Post比Get更安全）
* 需要传输大文本内容的时候（Post请求对数据长度没有要求）

所以，爬虫也需要在这两点模拟浏览器发送Post请求

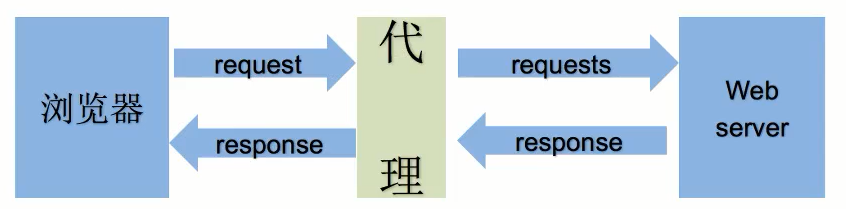
* 用法：

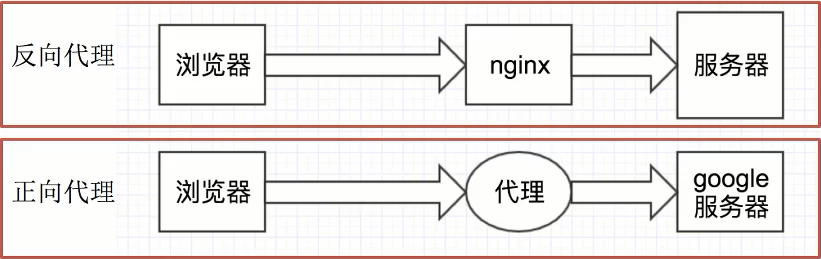
Response = requests.post([http://www.baifu.com/”,data.headers](http://www.baifu.com/) = headers)

data 的形式：字典



## 使用代理







## 为什么要使用代理？

* 让服务器以为不是同一个客户端在请求
* 防止我们的真实地址被泄露，防止被追究
* 准备IP地址，组成IP池（随机选择，排序随记优先级\*使用次数\*）

{“ip”:ip,“times”:0}

[{},{},{},{},{}]，对这个IP列表进行排序，按照使用次数

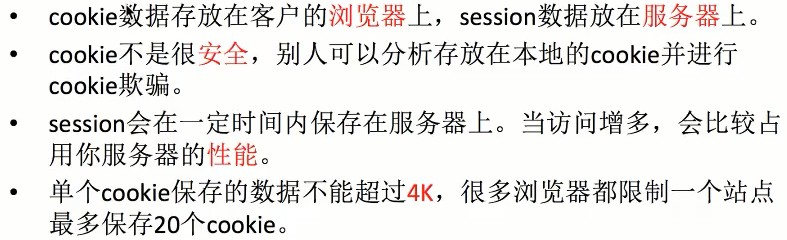
选择使用次数较少的几个IP，从中随机选择一个

## 检查IP的可用性

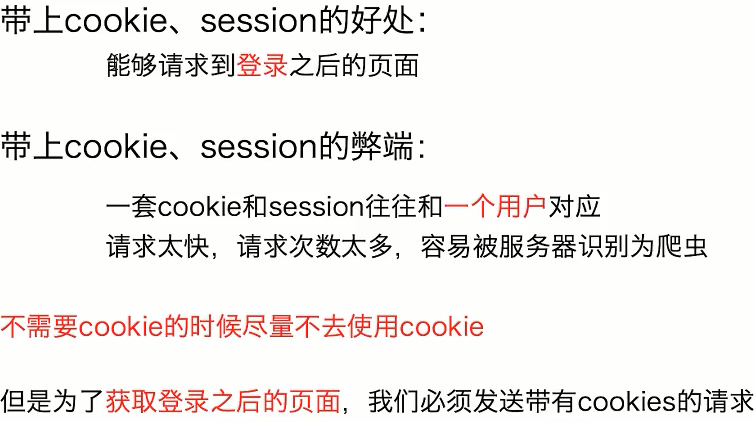
* 可以添加requests添加超时参数，判断IP地址的质量
* 在线代理IP质量检测的网站

## Reques模拟登录的3种方式

## Cookie和session的区别:



## 爬虫处理cookie和session



## 携带cookie进行请求

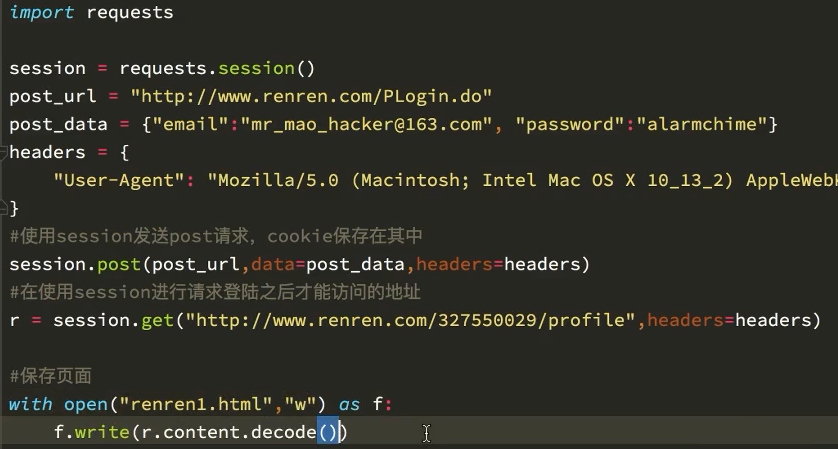
* 携带一堆cookie进行请求，把cookie组成cookie池

## 处理cookie、session请求



## 请求登录之后的网站思路

* 实例化session
* 先使用session发送请求，登录对应网站，把cookie保存在session中
* 再使用session请求登陆之后才能访问的网站，把session能够自动的携带登录成功是保存在其中的cookie，再进行请求



## 不发送POST请求，使用cookie获取登录后的页面

* Cookie过期时间很长的的网站
* 在cookie过期之前能够拿到所有数据
* 与获取cookie的其他工具配合

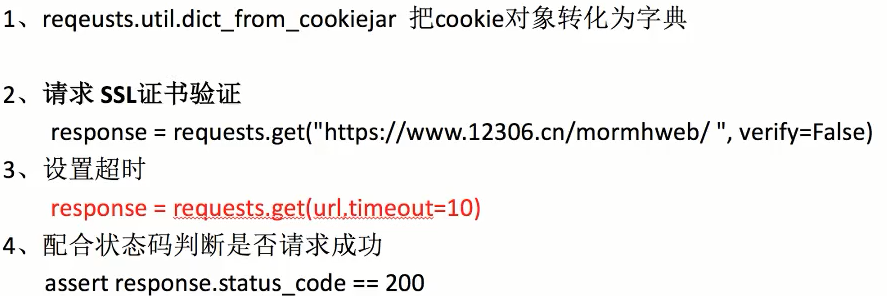
## 三种方式

* 实例化session，使用session发送post请求，再使用他获取登录后的页面
* Headers方法中添加cookie值，值为cookie字符串
* 在请求方法中添加cookies参数，接收字典形式的cookie。字典形式的cookie中的键是cookie的name，值是cookie的value

## 寻找登录的POST地址

* 在form表单中寻找action对应的url地址
  + POST的数据是input标签中的name作为键，真正的用户名密码作为值的字典。
* 抓包

## Requests小技巧



# 数据的提取

## 数据分类

* 结构化数据：josn、xml等
* 处理方法：转化为Python数据类型



* 非结构化数据：html等
* 处理方法：正则表达式、xpath



## 数据提取之josn

