第 1 章

网络爬虫概述

本章学习目标

- 了解什么是网络爬虫。
- 了解网络爬虫的搜索和策略。
- 了解反网络爬虫技术及解决方案。

本章先向读者介绍网络爬虫技术的概念和流程，再介绍数据采集的基本思想及网络爬虫的搜索和策略，最后介绍反网络爬虫技术及解决方案。

1.1 网络爬虫简介

1.1.1 网络爬虫的概念与类别

1. 概念

当前社会已经迈入大数据时代，互联网中的数据是海量的，如何自动高效地获取互联网中有用的信息是一个重要问题，而网络爬虫技术就是为解决这些问题而生的。

当下的网络就像一张大型的蜘蛛网，分布于蜘蛛网各个节点的即是数据，那么 Web Crawler（网络爬虫）即是小蜘蛛，沿着网络“捕获”食物（即数据），而网络爬虫是指按照一定的规则，自动地抓取网络信息的程序或者脚本。从专业角度来讲，请求目标的行为是经由程序模仿搜索引擎发出的，爬到本地的是目标返回的 HTML 代码、JSON 数据、二进制数据、图片、视频等，从中提取需要的数据并存储起来使用。

2. 常见的网络爬虫

搜索引擎如何获得一个新网站的 URL？主要描述如下：

（1）主动向搜索引擎提交网站。

（2）在其网站里设置外联。

（3）搜索引擎会和 DNS 服务商进行合作，能够快速采集新的网站。

常见的网络爬虫有以下几种。

- 通用网络爬虫：也叫全网爬虫，主要为门户网站站点搜索引擎和大型 Web 服务提供商采集网络数据。

通用网络爬虫并不是一切皆可爬取，它也要遵循 Robots 协议。通用网络爬虫的工作流程为：抓取网页→存储数据→内容处理→提供检索→排名服务。

通用网络爬虫的缺点有：仅提供与文本相关的内容（如 HTML、Word、PDF 等），而无法提供多媒体文件（如音乐、图片、视频）和二进制文件，提供的结果一成不变，无法针对不同背景领域的人提供不同的搜索结果，不能提供人类语义上的检索；具有局限性，所返回的网页里 90%的内容无用，中文搜索引擎的自然语言检索理解困难，信息占有量和覆盖率存在局限，以关键字搜索为主是搜索引擎最主要的作用之一，对于图片、数据库、音频、视频多媒体的内容无计可施；搜索引擎的社区化和个性化欠缺，大多数搜索引擎无法考虑人的地域、性别、年龄的差别，且抓取动态网页效果不好。

- 聚焦网络爬虫：网络爬虫程序员写的针对某种内容的网络爬虫，面向主题网络爬虫、面向需求网络爬虫，会针对某种特定的内容去抓取信息，并且保证内容需求尽可能相关。聚焦网络爬虫是为解决通用网络爬虫的缺点而出现的。
- 积累式网络爬虫：从头到尾，不断抓取，过程中会进行反复操作。
- 增量式网络爬虫：采用增量式更新和仅抓取新产生的或者已经发生变化的网页的网络爬虫，出现在已下载的网页。
- 深层网络爬虫：Web 页面按存在方式可以分为表层网页（Surface Web）和深层网页（Deep Web）。表层网页是指传统搜索引擎可以索引的页面，以超链接可以到达的静态网页为主构成的 Web 页面。深层网页是指那些大部分内容不能通过静态链接获取的、隐藏在搜索表单后的，只有用户提交一些关键词才能获得的 Web 页面。

3. 增量式网络爬虫

增量式网络爬虫（Incremental Web Crawler）的体系结构包括爬行模块、排序模块、更新模块、本地页面集、待爬行 URL 集及本地页面 URL 集。

增量式网络爬虫为确保本地页面集中存储的页面是最新页面并进一步提高本地页面集中页面的质量，经常使用的方法有：

（1）统一更新法：网络爬虫以相同的频率访问全部网页，不考虑网页的改变频率。

（2）个体更新法：网络爬虫按照个体网页的改变的频率重新访问各页面。

（3）基于分类的更新法：网络爬虫根据网页改变的频率把页面划分为更新较快网页子集和更新较慢网页子集两类，而后以不同的频率访问这两类网页。

为实现个体更新法，增量式网络爬虫需要对网页的重要性排序，一般常用的策略有广度优先策略、PageRank 优先策略等。IBM 开发的 WebFountain 增量式网络爬虫功能强大，它采用一个优化模型控制爬行过程，并没有对页面变动过程做一概统计假设，而是按照先前爬行周期里的爬行成果和网页的实际变化速度对页面的更新频率进行调整，采用的是一种自适应的方法。北京大学的天网增量爬行系统的目的是爬行国内 Web，把网页划分为变化网页和新网页两类，分别采用不同爬行策略。为减缓对大量网页变化历史维护导致的性能难题，它按照网页变化的时间局部性规律，在短时期内直接爬行屡次变化的网页，为尽快取得新网页，它操纵索引型网页追踪新出现的网页。

4. 深层网络爬虫

1）基于领域知识的表单填写

此方式一般会维持一个本体库，经由语义分析来选取合适的关键词填写表单。利用一个预定义的领域本体知识库来识别深层网页的页面内容，同时利用一些来自 Web 站点的导航模式来辨认主动填写表单时所需进行的路径导航。

2）基于网页结构分析的表单填写

此方式通常无领域知识或有唯一有限的领域知识，把网页表单表示成 DOM 树，从中提取表单各字段值。Desouky 等提出一种 LEHW 方法。该方法把 HTML 网页表示为 DOM 树形式，把表单区分为单属性表单和多属性表单，分别进行处理；孙彬等提出一种基于 XQuery 的搜索系统，它能够模拟表单和特殊页面的标记切换，把网页关键字切换信息描述为三元组单元，依照一定规则排除无效表单，把 Web 文档构造成 DOM 树，利用 XQuery 把文字属性映射到表单字段。

1.1.2　网络爬虫的流程

1. 基本流程

搜索引擎抓取系统的主要组成部分是网络爬虫，把互联网上的网页下载到本地形成一个联网内容的镜像备份是网络爬虫的主要目标。

网络爬虫的基本工作流程如下：一开始选取一部分精心挑选的种子 URL；把这些 URL 放入 URL 队列中；从 URL 队列中取出待抓取的 URL，读取 URL 之后开始解析 DNS，并把 URL 对应的网页下载下来，放进网页库中。此外，把这些 URL 放入已抓取 URL 队列。

分析已抓取 URL 队列中的 URL，并且把 URL 放入待抓取 URL 队列，使其进入下一个循环。网络爬虫的基本流程如图 1-1 所示。

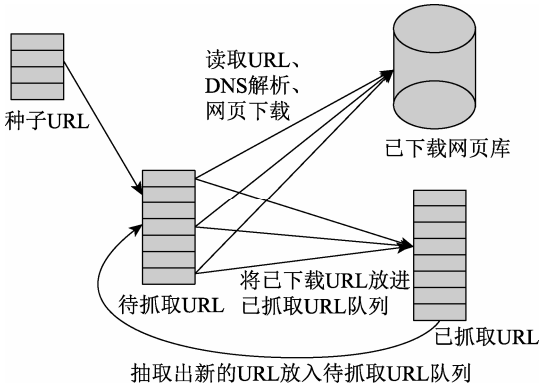

图 1-1　网络爬虫的基本流程

用简短易懂的方式来讲，即分为四个步骤：发送请求→获取响应内容→解析内容→保存数据。请求流程如图 1-2 所示。

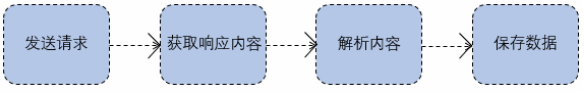

图 1-2　请求流程图

2. 从网络爬虫的角度对互联网进行划分

从网络爬虫的角度可将互联网划分为以下五种：

（1）已下载未过期网页。

（2）已下载已过期网页：抓取到的网页实际上是互联网内容的一个镜像与备份，互联网是动态变化的，一部分互联网上的内容已经发生变化，这时这部分抓取到的网页就已经失效。

（3）待下载网页：是指待抓取 URL 队列中的那些页面。

（4）可知网页：尚未抓取下来，也没有在待抓取 URL 队列中，但是能够经由对已抓取页面或者待抓取 URL 对应页面进行分析获得的 URL，认为是可知网页。

（5）不可知网页：还有一部分网页，网络爬虫是无法直接抓取下载的，称为不可知网页。

网页类别划分如图 1-3 所示。

3. 网页抓取的基本原理

常见的叫法是网页抓屏（Screen Scraping）、数据挖掘（Data Mining）、网络收割（Web Harvesting）或其类似的叫法。

理论上，网页抓取是一种经由多种方法收集网络数据的方式，不仅是经由与 API 交互的方式。

最常用的方法是确定爬取的 URL，确定数据存储格式，写一个自动化程序向网络服务器请求数据（通常是用 HTML 表单或其网页文件），而后对数据进行清洗解析，汲取需要的信息并存入数据库，基本思路如图 1-4 所示。

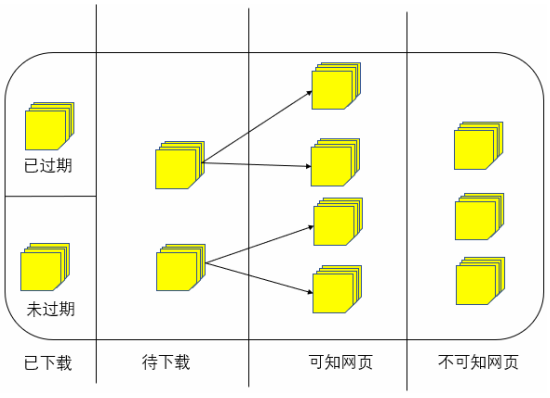

图 1-3　网页划分类别

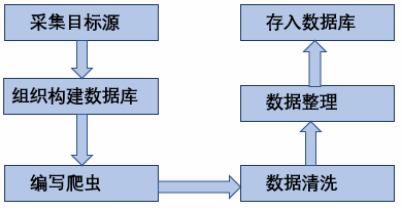

图 1-4　基本思路图

4. 目标源选择

目标源选择应依照以下条件进行排序：数据相关性、易抓取程度、数据量、Robots 协议。当然，根据自己的需求能够自由变更。同等情况下尽量避免大型企业的官网，因为其中大部分都设有反爬机制。

5. 编辑网络爬虫

推荐使用的库有 requests、BeautifulSoup、Scrapy、Selenium，假如关于效率需求不是特别高，能够考虑使用 requestspost 请求采集页面，而后使用 BeautifulSoup 分析页面标签，这样实现较为简短易懂，也能解决大部分需求；假如对效率比较重视，或需要完成一个工程化的采集项目，Scrapy 能够作为首选。对分布式处理的良好支持和清晰的模块化层次在提升效率的同时更易于进行代码的管理。对 HTTP 的相关请求，使用 requests 比用其他函数更加明智。

6. 数据清洗

获得的数据和期望中的数据总有一定的差别，这一部分的任务便是排除异常数据，把其余数据转换为易于处理的形式。数据的异常主要包括数据格式异常和数据内容异常。需要的数据可能存放在一个 PDF、Word、JPG 格式的文件中，把它们转换成文本而后选取相应的信息，这是数据清洗工作的一部分。另外，由于网页发布者的疏忽，网页上有部分数据和其他页面呈现不同，但需要把这部分数据也抓取下来，此时需要进行一定的处理，把数据格式进行统一。

1.1.3　网络爬虫的抓取

1. 概述

网络爬虫的不同抓取策略，便是利用不同的方法确定待抓取 URL 队列中 URL 的优先顺序。网络爬虫的抓取策略有很多种，但不管方法如何，其根本目标一致。网页的重要性评判标准不同，大部分采用网页的流行性进行定义。网页结构分布图如图 1-5 所示。

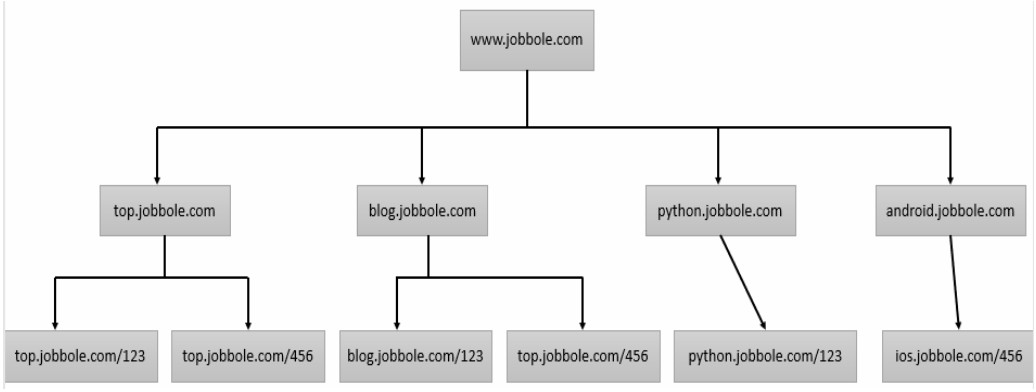

图 1-5　网页结构分布图

2. 网络爬虫的抓取原理

一开始选取一部分精心挑选的种子 URL，把这些 URL 放入待抓取 URL 队列，从待抓取 URL 队列中拿出待抓取的 URL，解析 DNS 并且得到主机的 IP 地址，并把 URL 相应的网页下载下来，存放进已下载网页库中。此外，把这些 URL 放进已抓取 URL 队列。分析已抓取 URL 队列中的 URL，分析当中的其他 URL，并且把 URL 放入待抓取 URL 队列，继续进入下一个循环。

3. 网络爬虫的抓取策略

1）宽度优先遍历（Breath First）策略

基本思路：将新下载网页包含的链接直接追加到待抓取 URL 队列末尾。

倘若网页是 1 号网页，从 1 号网页中抽取出 3 个链接指向 2 号、3 号和 4 号网页，于是按照编号顺序依次放入待抓取 URL 队列，图中网页的编号便是在待抓取 URL 队列中的顺序编号，之后网络爬虫以此顺序进行下载。抓取节点树结构如图 1-6 所示。

2）非完全 PageRank（Partial PageRank）策略

基本思路：对于已下载的网页，加上待抓取 URL 队列中的 URL 一起，形成网页集合，在此集合内进行 PageRank 计算，计算完成后，把待抓取 URL 队列里的网页依照 PageRank 得分由高到低排序，形成的序列便是网络爬虫接下来应该依次抓取的 URL 列表。

设定每下载 3 个网页进行新的 PageRank 计算，此时已经有{1,2,3}3 个网页下载到本地。这三个网页包含的链接指向{4,5,6}，即待抓取 URL 队列，如何决定下载顺序？将这 6 个网页形成新的集合，对这个集合计算 PageRank 的值，这样 4、5、6 就获得对应的 PageRank 值，由大到小排序，即可得出下载顺序。假设顺序为 5、4、6，当下载 5 号页面后抽取出链接，指向页面 8，此时赋予 8 临时 PageRank 值，如果这个值大于 4 和 6 的 PageRank 值，则接下来优先下载页面 8，如此不断循环，即形成非完全 PageRank 策略的计算思路。非完全 PageRank 策略结构图如图 1-7 所示。

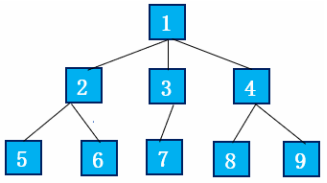

图 1-6　抓取节点树结构

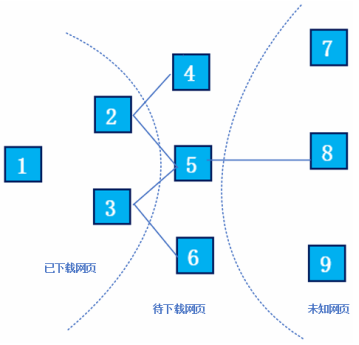

图 1-7　非完全 PageRank 策略结构图

3）OPIC（Online Page Importance Computation，在线页面重要性计算）策略

基本思路：在算法开始之前，每个互联网页面都给予相同的“现金”，每当下载某个页面后，此页面就把本身具有的“现金”平均分配给页面中包含的链接页面，把本身的“现金”清空。与 PageRank 的不同在于：PageRank 每次需要迭代计算，而 OPIC 策略不需要迭代过程。所以，OPIC 的计算速度远远快于 PageRank，适合实时计算使用。

4）大站优先（Larger Sites First）策略

基本思路：以网站为单位来选题网页重要性，关于待抓取 URL 队列中的网页，按照所属网站归类，假如哪个网站等待下载的页面最多，则优先下载这些链接，其本质思想倾向于优先下载大型网站，因为大型网站常常包括更多的页面。鉴于大型网站往往是著名企业的内容，其网页质量一般较高，所以这个思路虽然简单，但是有可靠依据。实验表明，这个算法结果也要略优先于宽度优先遍历策略。

1.2　网络爬虫的攻防战

网络爬虫是模仿人的浏览访问行为，进行数据的批量抓取。当抓取数据量慢慢增大时，会对被访问的服务器造成很大的压力，甚至有可能会崩溃。服务器第一种识别网络爬虫的方式便是经由检查连接的用户代理（User-Agent）来识别到底是浏览器访问，还是代码访问。假如是代码访问，当访问量增大时，服务器就会直接封掉来访 IP 地址。

在进行访问时，在开发者环境下不仅能够找到 URL、FormData，还能够在 requests 传入 headers 中构造浏览器的请求头封装，只需要构造这个请求头的参数，创建请求头部信息便可，代码如下：

```
    import requests
    headers = {
       'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/74.0.3729.157
Safari/537.36',
    }
    #经由 requests()方法构造一个请求对象
    url = r'https://www.baidu.com//'
    response = requests.get(url, headers = headers)
```

很多人会认为修改 User-Agent 太简短易懂，确实很简短易懂，但正常人一秒看一张图，而网络爬虫一秒能看几百张图，那么服务器的压力必然增大。也就是说，假如在一个 IP 地址下批量访问下载图片，这个行为不符合正常人类的行为，肯定会被限制。其原理也很简单易懂，便是统计每个 IP 地址的访问频率，此频率超过阈值，就会返回一个验证码，假如真的是用户访问，用户就会填写，而后继续访问，假如是代码访问，就会被限制。

这个问题的解决方法有两个，第一个便是常用的增设延时，每三秒抓取一次，代码如下：

```
    import time
    time.sleep(3)
```

其实，还有一个更重要的方法，那便是从本质解决问题。不管如何访问，服务器的目的都是查出哪些为代码访问，而后加以限制。解决办法如下：为以防无法访问，在数据采集之前经常会使用代理，可以通过设置 requests 的 proxies 属性的方式实现。

首先构建自己的代理 IP 地址池，把其以字典的形式赋值给 proxies，而后传输给 requests，代码如下：

```
    proxies = {
       "http":"http://10.10.1.10:3128",
       "https":"http://10.10.1.10:1080",
    }
    response = requests.get(url,proxies=proxies)
```

1.3　反网络爬虫技术及解决方案

1. 网络爬虫的危害

1）网络爬虫的影响

性能骚扰：Web 服务器默认接收人类访问，受限于编辑水平和目的，网络爬虫将会为 Web 服务器带来巨大的资源开销。

法律风险：服务器上的数据有产权归属，网络爬虫获取数据后牟利将带来法律风险。

2）侵犯用户隐私和知识产权

互联网用户的隐私、公司的商业机密，在大数据时代极易被网络爬虫窃取，相关的网络安全技术人员得采用必要的手段反爬，例如 cookie 反爬机制等，避免因商业机密的泄露而造成的经济损失。

客户端向服务器发送请求，请求头里面携带 cookie 信息，服务器检查 cookie 时效，如果 cookie 没有过期，则返回响应的请求数据。携带 cookie 发送请求如图 1-8 所示。

图 1-8　携带 cookie 发送请求

2. 反爬技术

1）验证码（Verification Code）

验证码是一种直接而有效的方式，用来判断请求方是否是人类。从一开始的简短易懂数字验证码，到后来的中文验证码，再到现在的图片验证码，验证码是应用层最普遍、最核心的网络爬虫对抗技术。关于一些简短易懂的数字、字母验证码，随着近几年机器学习、神经网络的快速发展，已经近乎于失效。有人训练出基于 LSTM 的模型能够达到 90%的识别正确率。关于图片验证码，也有专门用人工打码平台来处理，所以仅靠验证码很难有效处理网络爬虫问题，过多的验证码也会使正常用户的体验受到影响。简单验证码如图 1-9 所示。

图 1-9　简单验证码

2）Ajax

Ajax 技术在 2004 年左右开始飞速发展，成为主流的浏览器端技术，也使得网络爬虫从静态网络爬虫转化为动态网络爬虫。至此，抓取网站的数据不再是简短易懂的一个 HTTP 请求，解析 HTML 页面就能够实现的。大量的网站使用 Ajax 技术来构建网站，也使得解析数据变得相对没那么容易获取，因为在网站完全不设防的情况下，网络爬虫也不单需要解析 HTML 页面，同时还需要解析 Ajax 接口返回的数据，代码如下：

```
function get(){
    $.ajax({
        type:"OPTIONS",
        url:"/",
        compelete:function(d){
            var t = d.getResponseHeader("Date");
            var timestamp = Date.parse(t);
            var times = timestamp/1000;
            var dateNow = formateDate(new Date(timestamp));
            liveNow(dateNow,times);
        }
    });
};
```

3）数据混淆

网络爬虫的目标是抓取到有用的数据。对于许多应用来说，获取错误的数据往往比获取不

到数据更加致命。这个思路的核心便是，当网络爬虫命中反爬规则之后，使用错误的数据取代正确的数据返回给网络爬虫，这种方式十分隐蔽，又能够对敌手造成足够的麻烦，也相当有效。

4）经由 User-Agent 控制访问

不管是浏览器还是网络爬虫程序，在向服务器发起网络请求时，都会发过去一个头文件 headers，就像百度的请求头，大多数的字段都是浏览器向服务器表明身份用的，对于网络爬虫程序来说，最需要注意的字段便是 User-Agent。很多网站都会创建 User-Agent 白名单，只有属于正常范围的 User-Agent 才能够正常访问，代码如下：

```
#导入 requests 库
import requests
#定义获得 URL 的函数
def get_html(url):
    try:
       #请求获得 URL,超时时间 30s
     r=requests.get(url,timeout=30)
       #状态响应
     r.raise_for_status()
       #转换成 UTF-8 的编码格式
     r.encoding = r.apparent_encoding
       #返回响应的文本数据
     return r.text
    except:
       #运行错误返回值
        return "Something Wrong!"
#输出获得 URL
print(get_html('https://www.baidu.com'))
```

5）经由 IP 地址限制反网络爬虫

假如一个固定的 IP 地址在短暂的时间内快速大量地访问一个网站，那么自然会引起注意。管理员能够经由一些手段禁止该 IP 地址访问，网络爬虫程序则无法工作。

解决方法：比较成熟的方式是 IP 地址代理池，简而言之，便是经由 IP 地址代理，从不同的 IP 地址进行访问，这样就无法限制该 IP 地址的访问。但是 IP 地址代理的获得本身便是一个很麻烦的事情，虽然网上有免费和付费的，但是质量参差不齐。假如是企业需要，能够经由自己购买集群云服务来自建代理池。这里实现一个简短易懂的代理转换，代码如下：

```
import random
def get_proxy():
    '''
    简答模拟代理池
    返回一个字典类型的键值对,
'''
    proxy = ["http://116.211.143.11:80",
            "http://183.1.86.235:8118",
            "http://183.32.88.244:808",
            "http://121.40.42.35:9999",
            "http://222.94.148.210:808"]
    fakepxs = {}
    fakepxs['http'] = proxy[random.randint(0, len(proxy)-1)]
    return fakepxs
```

6）经由 Robots 协议限制网络爬虫

世界上将网络爬虫做得最大、最好的便是 Google。搜索引擎本身便是一个超级大的网络爬虫，Google 开发出来的网络爬虫 24 小时不中断地在网上抓取着新的信息，并返回给数据库，但是这些搜索引擎的网络爬虫都遵循着 Robots 协议。

Robots 协议是一种寄放于网站根目录下的 ASCII 编码的文本文件，它往往通知网络搜索引擎的漫游器，该网站中的哪些内容是不应被搜索引擎的漫游器获得的，哪些是能够被漫游器获得的。

Robots 协议并不是一个标准，而只是约定俗成的，所以并不能保护网站的隐私。注意，Robots 协议是用字符串比较来确定是否获得 URL，所以目录结尾有与没有斜杠“/”表示的是不同的 URL。Robots 协议允许使用类似'Disallow:*.gif'这样的通配符。

这实际上只是一个自由协议，遵循与否，都在于网络爬虫的编辑者。来看一下京东的 Robots 协议，代码如下：

```
User-agent: *
Disallow: /?*
Disallow: /pop/*.html
Disallow: /pinpai/*.html?*
User-agent: EtaoSpider
Disallow: /
User-agent: HuihuiSpider
Disallow: /
User-agent: GwdangSpider
Disallow: /
User-agent: WochachaSpider
Disallow: /
```

能够看到，京东的 Robots 协议里确定地指出四个“User-Agent”是禁止访问的，事实上，这四个 User-Agent 也是四个臭名远扬的恶性网络爬虫。当然有种情况是例外的，例如网络爬虫获得网页的速度和人类浏览网页是差不多的，这并不会给服务器造成太大的性能损失，在这种情况下是可以不用遵守 Robots 协议的。

1.4 本章习题

一、单选题

1. 网络爬虫的基本流程是（　　）。

A. 发送请求 → 获取响应内容 → 解析内容 → 保存数据

B. 发送请求 → 解析内容 → 获取响应内容 → 保存数据

C. 发送请求 → 获取响应内容 → 保存数据

D. 发送请求 → 解析 DNS → 获取响应内容 → 保存数据

2. 组织数据采集基本思想的第一步是（　　）。

A. 组织数据库　　B. 网络爬虫编写　　C. 数据整理　　D. 采集目标源

3. 以下选项中，（　　）不是爬行策略中的特征。

A. 脚本语言　　B. 巨大的数据量　　C. 快速的更新频率　　D. 动态页面的产生

4. 网络爬虫的系统框架中，（　　）不是主过程选择。

A. 服务器　　B. 控制器　　C. 解析器　　D. 资源库

5. 以下选项中，（　　）不是 Python requests 库提供的方法。

A. get()　　B. push()　　C. post()　　D. head()

6. 以下选项中，（　　）不是网络爬虫带来的负面问题。

A. 法律风险　　B. 隐私泄露　　C. 商业利益　　D. 性能骚扰

7. 如果一个网站的根目录下没有 robots.txt 文件，则以下说法中不正确的是（　　）。

A. 网络爬虫应该以不对服务器造成性能骚扰的方式抓取内容

B. 网络爬虫可以不受限制地抓取该网站内容并进行商业使用

C. 网络爬虫可以肆意抓取该网站内容

D. 网络爬虫的不当抓取行为仍然具有法律风险

二、简答题

1. 什么是网络爬虫？

2. 简述网络爬虫的基本流程。

3. 列举三种网络爬虫策略，并简单说明原理。

4. 列举三种常见反网络爬虫技术，并简单说明。

第 2 章

Python 基本知识介绍

本章学习目标

- 掌握 Python 的安装与环境配置。
- 掌握 PyCharm 的使用。
- 了解 HTML 的内容。
- 掌握 Python 基本库的使用。

本章先向读者介绍 Python 的安装与环境配置，再介绍 HTML 及其基本原理，最后介绍 Python 基本库的使用。

2.1 Python 编程

2.1.1 Python 的安装与环境配置

Python 是一门计算机编程语言。相比于 C 语言及 Java 来说，Python 更容易上手，同时也十分简单，易懂易用。很多大型网站，例如 YouTube、Google 等都在大量使用 Python，各种常用的脚本任务用 Python 实现也十分容易，无须担心学非所用。

1. 计算机编程语言这么多，为什么用 Python 来写网络爬虫呢？

（1）对比其他静态编程语言来说，如 Java、C#、C++，Python 抓取网页文档接口更加简洁；对比其他动态语言 Perl、Shell，Python 的 urllib2 包提供非常完整的访问网页文档 API。抓住网页有时候需要模拟浏览器的行为，而 Python 具有很多第三方包，如 requests、XPath 等均提供此类支持。

（2）对于抓取之后的网页需要进行处理，如过滤标签、提取文本等。Python 提供简洁的文档处理功能，可以用很短的代码完成大部分文档处理。

（3）具有各种网络爬虫框架，可方便高效地下载网页。

（4）多线程、进程模型成熟稳定，网络爬虫是一个典型的多任务处理场景，请求页面时会有较长的延迟，总体来说更多的是等待。多线程或进程会更优化程序效率，提升整个系统的下

载和分析能力。

2. Python 的概念

百度百科解释 Python 是一种计算机程序设计语言，是一种面向对象的动态类型语言，最初被设计用于编辑自动化脚本(Shell)，随着版本的不断更新和语言新功能的添加，Python 被越来越多地用于独立的、大型项目的开发。

Python 是有名的“龟叔”Guido van Rossum 在 1989 年圣诞节期间，为打发无聊的圣诞节而编辑的一个编程语言。

Python 能够提供十分完善的基础代码库，涵盖网络、文件、GUI、数据库、文本等大量内容，被形象地称作“内置电池”。用 Python 开发，很多功能没必要从零编辑，直接利用现有的即可。

除内置的库外，Python 还有大量的第三方库，也就是别人开发的、供直接运用的工具。当然，假如开发的代码经由很好的封装，也能够作为第三方库给别人使用。

很多大型网站都是用 Python 开发的，例如 YouTube、Instagram，还有国内的豆瓣网。很多大公司，包括 Google、Yahoo 等，乃至 NASA（美国航空航天局）都大量地应用 Python。

“龟叔”赋予 Python 的定位是“优雅”“明确”“简短易懂”，所以 Python 程序看上去总是简短易懂。初学者学习 Python，不但初学容易，而且来日深入下去，能够编辑那些十分复杂的程序。

总的来说，Python 的哲学便是简短、易懂、优雅，尽可能写出容易看明白的代码，尽可能写少的代码。

3. Python 的应用领域

Python 拥有很多免费数据函数库、免费 Web 网页模板系统和与 Web 服务器进行交互的库，能够实现 Web 开发，搭建 Web 框架，目前比较有名气的 Python Web 框架为 Django。同样是解释型语言的 JavaScript，在 Web 开发中的应用已经较为广泛，原因是其有一套完善的框架。但 Python 也有着特有的优势。例如，Python 相比于 JS、PHP 在语言层面较为完好并且关于同一个开发需求能够提供多种方案，库的内容丰富，使用方便。从事该领域应从数据、组件、安全等多领域进行学习，从底层了解其工作原理并可支配任何业内主流的 Web 框架。

下面来介绍一下基于 Python 语言的 Web 开发中几种常见的 Web 开发框架。

1）Django

Django 是一个常见的 Python Web 应用框架。它是开源的 Web 开发框架，包括多种组件，能够实现关系映射、动态内存管理、界面管理等功能。Django 开发采用 DRY 原则，同时拥有独立的轻量级 Web 服务器，能快速开发 Web 应用。Django 开发遵循 MVC 模式，包括模型、视图、控制三部分。模型层是应用程序底层，主要用于处理与数据有关的事件，如数据存取验证等。由于 Django 中用户输入控制模块是由框架处理的，因此也能够称为模板层。模板层用于展现数据，负责模板的存取和正确调用模板等业务。程序员使用模板语言来渲染 HTML 页面，给模板所需显示的信息，使用既定的模板来渲染结果。视图层组成应用程序的业务逻辑，负责在网页或类似类型的文档中展示数据。

2）CherryPy

CherryPy 是基于 Python 的面向对象的 HTTP 框架，适合 Python 开发者。CherryPy 本身内置 Web 服务器。CherryPy 的用户无须搭设别的 Web 服务器，能直接在内置的服务器上运行应用程序。服务器一方面把底层 TCP 套接字传输的信息转换成 HTTP 请求，并传递给相应的处理

程序；另一方面把上层软件传来的信息打包成 HTTP 响应，向下传递给底层的 TCP 套接字。

3）Flask

Flask 适合开发轻量级的 Web 应用。它的服务器网关接口工具箱采用 Werkzeug，模板引擎使用 Jinja2。Flask 使用 BSD 授权。Flask 自身没有如表单验证和数据库抽象层等一些基本功能，而是依附第三方库来完成这些工作。Flask 的结构是可扩展的，能够比较容易地为它添加一些需要的功能。

4）Pyramid

Pyramid 是开源框架，执行效率高，开发周期短。Pyramid 包含 Python、Perl、Ruby 的特性，拥有不依赖于平台的 MVC 架构，以及最快的启动开发的能力。

5）TurboGear

TurboGear 创建在其框架基础上，它尝试把其框架优秀的部分集成到一起。它允许开发者从一个单文件服务开始开发，慢慢扩大为一个全栈服务。

4. 数据分析与处理

通常情况下，Python 被用来做数据分析。用 C 设计一些底层的算法进行封装，而后用 Python 进行调用。由于算法模块较为固定，所以用 Python 直接进行调用，方便且灵活，能够根据数据分析与统计的需要灵活使用。Python 也是一个比较完善的数据分析生态系统，其中，matplotlib 常常会被用来绘制数据图表，它是一个 2D 绘图工具，有着杰出的跨平台交互特性，日常做描述统计用到的直方图、散点图、条形图等都会用到它，几行代码便可出图。平常看到的 K 线图、月线图也可用 matplotlib 绘制。假如在证券行业做数据分析，Python 是必不可少的。

随着大数据和人工智能时代的到来，网络和信息技术开始渗透到人类日常生活的方方面面，产生的数据量也呈现指数级增长的态势，同时现有数据的量级已经远远超过目前人力所能处理的范畴。在此背景下，数据分析成为数据科学领域中一个全新的研究课题。在数据分析的程序语言选择上，由于 Python 语言在数据分析和处理方面的优势，大量的数据科学领域的从业者使用 Python 来进行数据科学相关的研究工作。

数据分析是指用合适的分析方法对收集来的大量数据进行分析，提取实用信息和构成结论，对数据加以具体钻研和归纳总结的过程。随着信息技术的高速发展，企业的生产、收集、存储数据的能力大大提升，同时数据量也与日俱增。把这些繁杂的数据经由数据分析方法进行提炼，以此研究出数据的发展规律和展望趋向走向，进而帮助企业管理层做出决策。

数据分析是一种解决问题的过程和方法，主要的步骤有需求分析、数据获得、数据预处理、分析建模、模型评价与优化、部署。下面分别介绍每个步骤。

1）需求分析

数据分析中的需求分析是数据分析环节中的第一步，也是十分重要的一步，决定后续的分析方法和方向。主要内容是根据业务、生产和财务等部门的需要，结合现有的数据情况，提出数据分析需求的整体分析方向、分析内容，最终和需求方达成一致。

2）数据获得

数据获得是数据分析工作的基础，是指按照需求分析的结果提取、收集数据。数据获得主要有两种方式：网络爬虫获得和本地获得。网络爬虫获得是指经由网络爬虫程序合法获得互联网中的各种文字、语音、图片和视频等信息；本地获得是指经由计算机工具获得存储在本地数据库中的生产、营销和财务等系统的历史数据和实时数据。

3）数据预处理

数据预处理是指对数据进行数据合并、数据清洗、数据标准化和数据变换，并直接用于分析建模的这一过程的总称。其中，数据合并能够把多张互相关联的表格合并为一张；数据清洗能够去掉重复、缺失、异常、不一致的数据；数据标准化能够去除特征间的量纲差异；数据交换则能够经由离散化、哑变量处理等技术满足后期分析与建模的数据要求。在数据分析过程中，数据预处理的各个过程互相交叉，并没有固定的先后顺序。

4）分析建模

分析建模是指经由对比分析、分组分析、交叉分析、回归分析等分析方法，以及聚类模型、分类模型、关联规则、智能推荐等模型和算法，发现数据中的有价值信息，并得出结论的过程。

5）模型评价与优化

模型评价是指对于已经创建的一个或多个模型，根据其模型的类型，使用不同的指标评价其性能好坏的过程。模型的优化则是指模型性能在经由模型评价后已经达到要求，但在实际生产环境应用过程中，发现模型的性能并不理想，继而对模型进行重构与优化的过程。

6）部署

部署是指把数据分析结果与结论应用至实际生产系统的过程。根据需求的不同，部署阶段可以是一份包含现状具体整改措施的数据分析报告，也可以是把模型部署在整个生产系统的解决方案。在多数项目中，数据分析员提供的是一份数据分析报告或者一套解决方案，实际执行与部署的是需求方。

Python 是一门应用十分广泛的计算机语言，在数据科学领域具有无可比拟的优势。Python 正在逐步成为数据科学领域的主流语言。Python 数据分析具备以下几方面优势：

（1）语法简短、易懂、精练。对于初学者来说，比起其他编程语言，Python 更容易上手。

（2）有很多功能强大的库。结合在编程方面的强大实力，只使用 Python 这一种语言就能够去构建以数据为中心的应用程序。

（3）不单适用于研究和构建原型，同时也适用于构建生产系统。研究人员和工程技术人员使用同一种编程工具，能给企业带来明显的组织效益，并降低企业的运营成本。

（4）Python 程序能够以多种方式轻易地与其语言的组件“粘接”在一起。例如，Python 的 C 语言 API 能够帮助 Python 程序灵活地调用 C 程序，这意味着用户能够根据需要给 Python 程序添加功能，或者直接使用 Python 语言，不要调用 API 接口。

（5）Python 是一个混合体，丰富的工具集使它介于系统的脚本语言和系统语言之间。Python 不但具备全部脚本语言简短易懂和易用的特点，还拥有编译语言所具有的高级软件工程工具。

Python 具有 IPython、NumPy、SciPy、Pandas、Matplotlib、Scikit-learn 和 Spyder 等功能齐全、接口统一的库，能为数据分析工作提供极大的便利。

5. 人工智能应用

人工智能的核心算法是完全依赖于 C/C++的，由于是计算密集型，因此需要十分致密的优化，还需要 GPU、专用硬件之类的接口，这些都只有 C/C++能做到。所以在某种意义上，其实 C/C++才是人工智能领域最主要的语言。Python 是这些库的 API Binding，使用 Python 是因为 C-Python 的胶水语言特性，要开发一个其语言到 C/C++的跨语言接口用 Python 是最容易的，比其语言门槛要低不少，尤其是使用 C-Python 的时候。

说到 AI，Python 已经逐步成为一些 AI 算法的一部分，从最开始的简短易懂的双人游戏到后来复杂的数据工程任务。Python 的 AI 库在现今的软件中充当着不可取代的角色，包括 NLYK、

PyBrain、OpenCV 和 AIMA，一些 AI 软件功能，短短的一个代码块就足够。再看人脸识别技术、会话接口等领域，Python 正在一步步覆盖更多新领域。可以说，Python 未来的潜力是不可估量的。

在人工智能的应用方面，例如在神经网络、深度学习方面，Python 都能够找到比较成熟的包来加以调用。并且 Python 是面向对象的动态语言，且适用于科学计算，这就使得 Python 在人工智能方面颇受喜爱。虽然人工智能程序不限于 Python，但仍然为 Python 提供大量的 API，这也正是由于 Python 当中包含着较多的适用于人工智能的模块，如 sklearn 模块等。调用方便、科学计算功能强大依旧是 Python 在 AI 领域最强大的竞争力。

6. Linux 系统下 Python 的安装

1）创建路径

首先创建一个 Pyhton 的安装路径，命令如下，创建安装路径结果如图 2-1 所示。

```
rm -rf /usr/local/python3
sudo mkdir /usr/local/python3
su root
chmod 777 /usr/local/python3
```

图 2-1　创建安装路径结果

创建完成后进入该文件夹中，命令如下，结果如图 2-2 所示。

```
cd /usr/local/python3
```

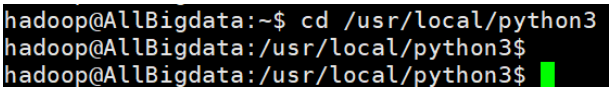

图 2-2　进入该文件夹

2）下载安装包

进入该文件夹后，下载安装包到当前路径文件夹中，Linux 系统能够使用 wget 命令来执行，代码如下，下载进度如图 2-3 所示。

```
wget --no-check-certificate https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz
```

3）解压安装包

在当前文件夹下解压，代码如下：

```
tar -xzvf Python-3.6.5.tgz
```

图 2-3　下载进度

4）编译安装

进入该 Python 3.6.5 文件中进行编译安装，代码如下：

```
#进入解压完的文件夹
cd Python-3.6.5
#编译
sudo ./configure --prefix=/usr/local/python3
#安装
make
make install
#如在编译过程报错可使用这条命令解决
sudo apt-get install build-essential
```

5）创建链接

在编译与安装完成后，再创建 Python 3 的软链接，代码如下：

```
sudo ln -s /usr/local/python3/bin/python3  /usr/bin/python3
```

这时会报错，显示 failed to create symbolic link '/usr/bin/python3':File exists：无法创建符号链接'/usr/bin/python3'：文件存在，这时使用 rm 删掉相同名称的链接便可，代码如下，创建软链接结果如图 2-4 所示。

```
rm -rf /usr/bin/python3
```

```
root@AllBigdata:/usr/local/python3/Python-3.6.5# rm /usr/bin/python3
root@AllBigdata:/usr/local/python3/Python-3.6.5# sudo ln -s /usr/local/python3/bin/p
ython3  /usr/bin/python3
root@AllBigdata:/usr/local/python3/Python-3.6.5#
```

图 2-4　创建软链接结果

6）测试 Python 3 是否可用

在终端窗口中输入 Python 3 查看是否可用。进入 Python 3 后如图 2-5 所示。

```
root@AllBigdata:/usr/local/python3/Python-3.6.5# python3
Python 3.6.5 (default, Jan 17 2022, 09:19:18)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 2-5　进入 Python 3

7）安装 setuptools

在安装 pip 之前，需要先安装 setuptools，它是一组 Python 的 distutilsde 工具的增强工具，能够让程序员更方便地创建和发布 Python 包，特别是那些对其他包具有依赖性的情况。当需要安装第三方 Python 包时，可能会用到 easy_install 命令。使用 easy_install 命令实际上是在调用 setuptools 来完成安装模块的工作。

首先下载软件包，与安装 Python 类似，使用 wget 来进行下载，代码如下，下载 setuptools 如图 2-6 所示。

```
cd /usr/local/python3
wget https://pypi.python.org/packages/source/s/setuptools/setuptools-19.6.tar.gz
```

```
root@AllBigdata:/usr/local/python3# cd /usr/local/python3
root@AllBigdata:/usr/local/python3# wget https://pypi.python.org/packages/source/s/s
etuptools/setuptools-19.6.tar.gz
--2022-01-17 09:35:17--  https://pypi.python.org/packages/source/s/setuptools/setupt
ools-19.6.tar.gz
正在解析主机 pypi.python.org (pypi.python.org)... 151.101.76.223, 2a04:4e42:12::223
正在连接 pypi.python.org (pypi.python.org)|151.101.76.223|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 301 Redirect to Primary Domain
位置：https://pypi.org/packages/source/s/setuptools/setuptools-19.6.tar.gz [跟随至新
的 URL]
--2022-01-17 09:35:18--  https://pypi.org/packages/source/s/setuptools/setuptools-19
.6.tar.gz
正在解析主机 pypi.org (pypi.org)... 151.101.64.223, 151.101.0.223, 151.101.192.223,
```

图 2-6　下载 setuptools

8）解压并安装 setuptools 工具，代码如下：

```
#解压软件
tar -zxvf setuptools-19.6.tar.gz
cd setuptools-19.6
#编译并安装
python3 setup.py build
python3 setup.py install
```

9）开始安装 pip

使用 wget 直接从官网拉取软件包，代码如下，下载 pip 包如图 2-7 所示。

```
cd /usr/local/python3
wget --no-check-certificate https://pypi.python.org/packages/source/p/pip/pip-10.
0.1.tar.gz
```

```
root@AllBigdata:/usr/local/python3# wget --no-check-certificate https://pypi.python.
org/packages/source/p/pip/pip-10.0.1.tar.gz
--2022-01-17 09:59:17--  https://pypi.python.org/packages/source/p/pip/pip-10.0.1.ta
r.gz
正在解析主机 pypi.python.org (pypi.python.org)... 151.101.76.223, 2a04:4e42:12::223
正在连接 pypi.python.org (pypi.python.org)|151.101.76.223|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 301 Redirect to Primary Domain
位置：https://pypi.org/packages/source/p/pip/pip-10.0.1.tar.gz [跟随至新的 URL]
--2022-01-17 09:59:17--  https://pypi.org/packages/source/p/pip/pip-10.0.1.tar.gz
正在解析主机 pypi.org (pypi.org)... 151.101.128.223, 151.101.192.223, 151.101.0.223,
 ...
正在连接 pypi.org (pypi.org)|151.101.128.223|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 301 Moved Permanently
位置：https://files.pythonhosted.org/packages/source/p/pip/pip-10.0.1.tar.gz [跟随至
新的 URL]
```

图 2-7　下载 pip 包

之后安装 pip 包，代码如下：

```
#下载完后进行解压
tar -zxvf pip-10.0.1.tar.gz
#解压完成后进入文件夹中进行编译和安装
cd /usr/local/python3/pip-10.0.1
python3 setup.py build
#安装 pip
sudo apt-get install python3-pip
```

10）测试 pip 是否安装成功

使用 pip 安装一个 Python 包，成功则表示 pip 安装成功，如报错或者没有显示，则可使用更新机制，代码如下，测试 pip 是否安装成功如图 2-8 所示。

```
#验证 pip3 安装是否成功
pip3 install ipython
```

```
#查看 pip3 的版本
pip3 -V
#安装 lxml 包
pip3 install lxml
#升级 pip
sudo pip3 install --upgrade pip
```

```
root@AllBigdata:/usr/local/python3/pip-10.0.1#
root@AllBigdata:/usr/local/python3/pip-10.0.1# pip3 -V
pip 9.0.3 from /usr/local/python3/lib/python3.6/site-packages (python 3.6)
root@AllBigdata:/usr/local/python3/pip-10.0.1#
root@AllBigdata:/usr/local/python3/pip-10.0.1#
root@AllBigdata:/usr/local/python3/pip-10.0.1# pip3 install lxml
```

图 2-8　测试 pip 是否安装成功

2.1.2　PyCharm 的安装与使用

PyCharm 是一种 Python IDE，带有一整套能够帮助用户在使用 Python 语言开发时提高效率的工具，如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。另外，该 IDE 能够提供一些高级功能，以用于支持 Django 框架下的专业 Web 开发。

1. 下载 PyCharm 安装包

首先要下载 PyCharm 的安装包，下载地址为 https://www.jetbrains.com/pycharm/download/#section=linux，PyCharm 官网如图 2-9 所示。

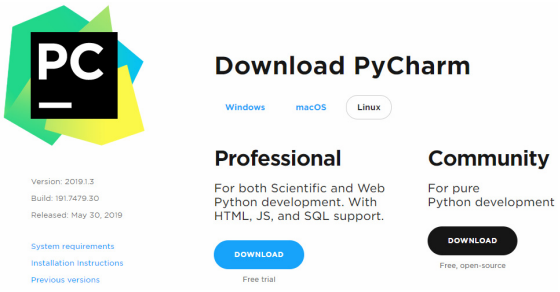

图 2-9　PyCharm 官网

2. 把该文件移动到 python 文件夹中解压并赋予权限

```
#进入安装目录
cd /usr/local/python3
#解压安装文件
tar -zxvf pycharm-professional-2021.3.1.tar.gz
#权限设置
chmod -R 777 /usr/local/python3/pycharm-2021.3.1
```

3. 安装前修改 hosts 文件

```
#在终端设备中输入命令
vi /etc/hosts
#在打开的文件中加入下面的代码
0.0.0.0 account.jetbrains.com
```

配置 hosts 如图 2-10 所示。

4. 进入 PyCharm 下的 bin 目录中，执行 sh 命令开始安装

```
#安装命令
cd /usr/local/python3/pycharm-2021.3.1/bin
export DISPLAY=localhost:0.0
sh ./pycharm.sh
```

在弹出的窗口中单击 OK 按钮。之后弹出 PyCharm Privary Policy Agreement 对话框，即隐私政策协议，直接单击 Continue 按钮同意。在弹出的发送请求中，单击发送统计信息。选择风格，单击 Next:Featuredplugins，再单击 StartusingPyCharm。

5. 搭建 Python 解释器

打开设置界面，配置 PyCharm 如图 2-11 所示。

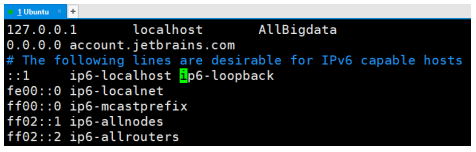

图 2-10 配置 hosts

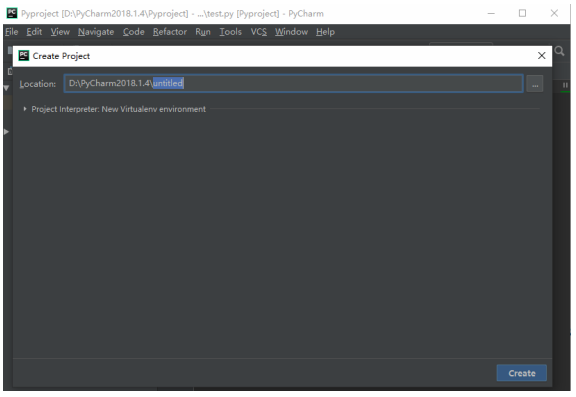

图 2-11 配置 PyCharm

搭建 Python 解释器，选择 Python 的安装路径后单击 OK 按钮确定，进入 Python 解释器配置，如图 2-12 所示，配置 Python 解释器路径如图 2-13 所示。

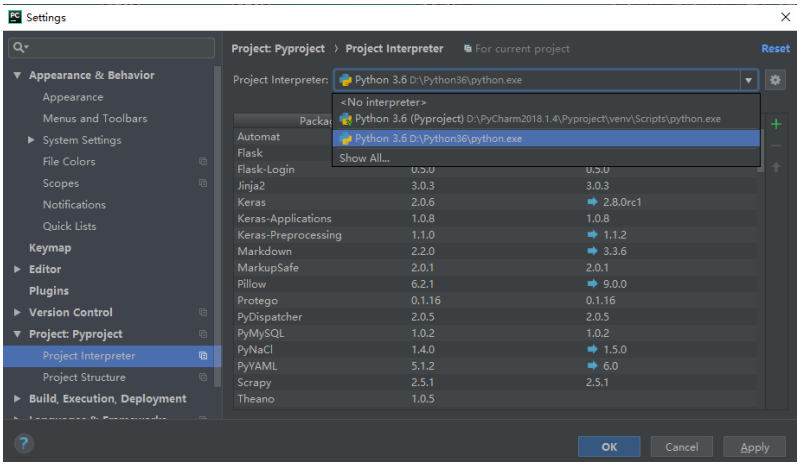

图 2-12 进入 Python 解释器配置

6. 创建项目与文件

打开 PyCharm 软件，单击创建新项目，选择存放的项目路径及名称后，选择建好的 Python 环境，配置 Python 解释器如图 2-14 所示。

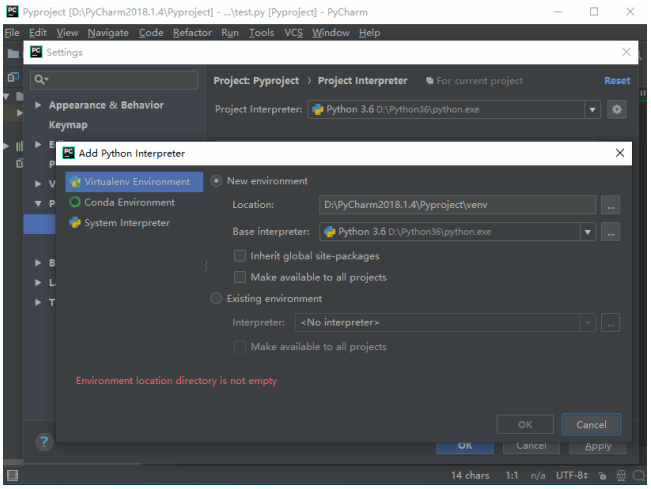

图 2-13　配置 Python 解释器路径

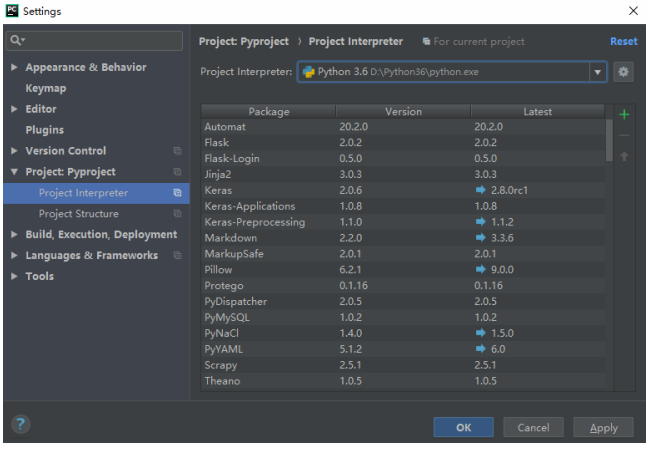

图 2-14　配置 Python 解释器

新建 Python 文件，右击项目名，选择 New→Python-File，输入代码。第一次运行右击编辑区域，单击 Run 命令，以后可直接单击右上角或者左下角的绿三角▶按钮，在 PyCharm 软件中运行 Python 如图 2-15 所示。

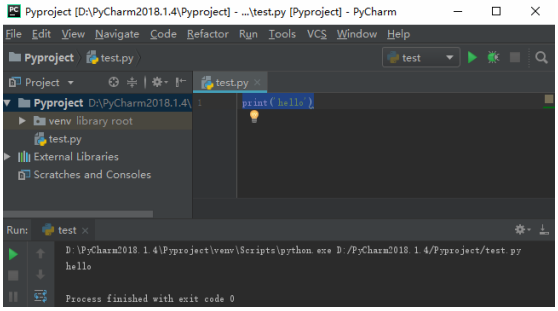

图 2-15　在 PyCharm 软件中运行 Python

到这里，Linux 系统的 PyCharm 就已经安装完成，Windows 系统下的 PyCharm 安装方法与此类似，不同是路径的选择及 host 文件部分。

2.2 HTML 基本原理

2.2.1 HTML 简介

1. HTML 解释

（1）HTML 是指超文本标记语言（Hyper Text Markup Language）。

（2）HTML 不是一种编程语言，而是一种标记语言（Markup Language）。

（3）标记语言是一套标记标签（Markup Tag）。

（4）HTML 使用标记标签来描述网页。

2. HTML 标签

（1）HTML 标签是由尖括号包围的关键词，如<html>。

（2）HTML 标签通常是成对出现的，如和。

（3）标签对中的第一个标签是开始标签，第二个标签是结束标签。

（4）开始标签和结束标签也被称为开放标签和闭合标签。

3. HTML 文档=网页

（1）HTML 的基本原理是 HTML 文档描述网页。

（2）HTML 文档包含 HTML 标签和纯文本。

（3）HTML 文档也被称为网页。

2.2.2 HTML 的基本原理

1. HTML

HTML 是超文本标记语言，不需要编译，直接经由浏览器执行，例如如下代码：

```
<input type= "text" name= "jake" />
```

2. 基本作用

（1）能够编辑静态网页，在网页显示图片、文字、声音、表格、链接。

（2）静态网页 html 只能够写成静态网页 html。

（3）动态网页是能够交互的，而不是动画。

HTML 发展史如图 2-16 所示。

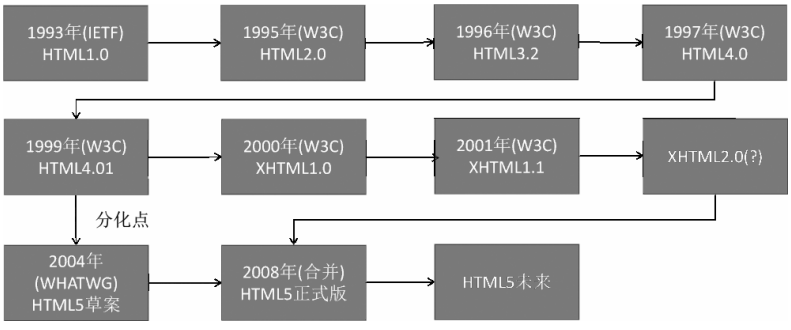

图 2-16 HTML 发展史

2.3　基本库的使用

2.3.1　urllib 库

在 Python 3 中，urllib 和 urllib2 进行归并，目前只有一个 urllib 模块，urllib 和 urllib2 中的内容整合进 urllib.request，urlparse 整合进 urllib.parse。

（1）urlparse 把 urlstr 解析成各个组件，代码如下：

```
#-*- coding:utf-8 -*-
import urllib.request
import urllib.parse
urlstr = "http://www.baidu.com"
parsed = urllib.parse.urlparse(urlstr)
print(parsed)
```

解析组件执行结果如图 2-17 所示。

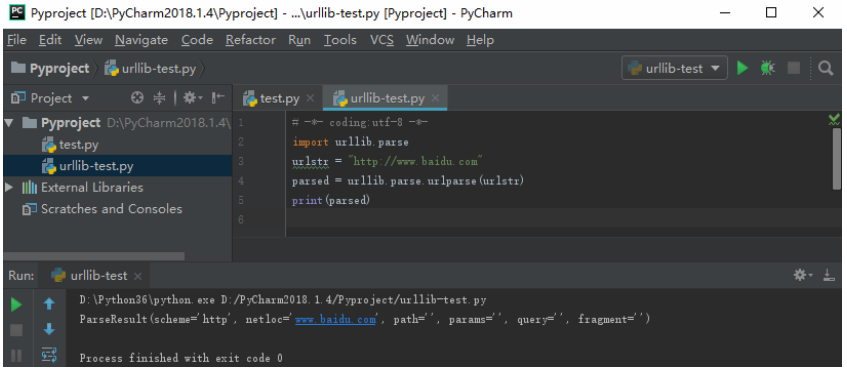

图 2-17　解析组件执行结果

（2）urljoin 把 URL 的根域名和新 URL 拼合成一个完整的 URL，代码如下：

```
import urllib.parse
url = "http://www.baidu.com"
new_path = urllib.parse.urljoin(url,"index.html")
print(new_path)
```

（3）urlopen 打开一个 URL 的方法，返回一个文件对象，而后能够进行类似文件对象的操作，代码如下：

```
import urllib.request
req = urllib.request.urlopen('http://www.baidu.com')
print(req.read())
```

2.3.2　requests 库

requests 库基于 urllib，且比 urllib 更加方便，是 Python 更加简短易懂的 http 库。以下是使用 requests 库的一个例子：

```
import requests
response = requests.get('http://www.baidu.com')
```

```
print(type(response))          #返回值的类型
print(response.status_code)    #当前网站返回的状态码
print(type(response.text))     #网页内容的类型
print(response.text)           #网页的具体内容（html 代码）
print(response.cookies)        #网页的 Cookie
```

requests 中输出网页的 HTML 代码的方法是 response 方法，它相当于 urllib 库的 response.read 方法，只不过不需要进行 decode 操作。打印 Cookie 的操作也比 urllib 简短易懂，只需要使用 Cookie 方法便可，各种请求方式代码如下：

```
import requests
requests.post('http://httpbin.org/post')
requests.delete('http://httpbin.org/delete')
requests.put('http://httpbin.org/put')
requests.head('http://httpbin.org/get')
requests.options('http://httpbin.org/get')
```

2.3.3 re 库

1. match()方法

从字符串头部开始匹配，代码如下：

```
import re
content = 'The123456ismyonephonenumber.'
print(len(content))    #字符串长度
result = re.match(r'^The',content)
                       #使用 match 匹配，第一个参数为正则表达式，第二个为要匹配的字符串
print(result)
print(result.group()) #输出匹配内容
print(result.span())   #输出匹配内容的位置索引
```

match 用法执行结果如图 2-18 所示。

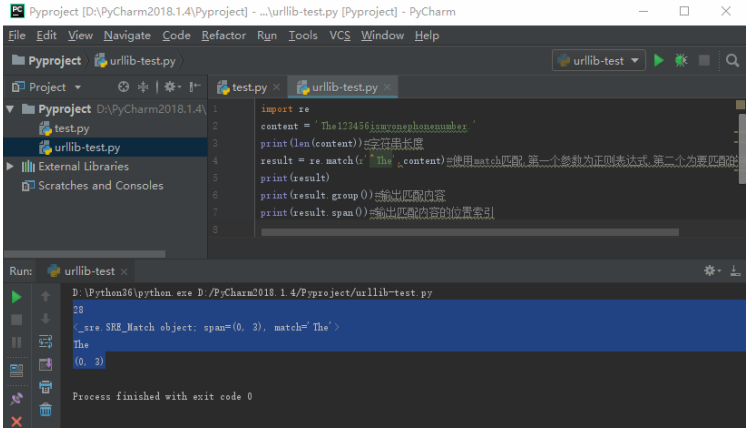

图 2-18　match 用法执行结果

2. 公用匹配

```
import re
content = 'The123456ismyonephonenumber.'
```

```
result = re.match(r'^The.*',content)
print(result)
print(result.group())  #输出匹配内容
print(result.span())   #输出匹配内容的位置索引
```

公用匹配用法如图 2-19 所示。

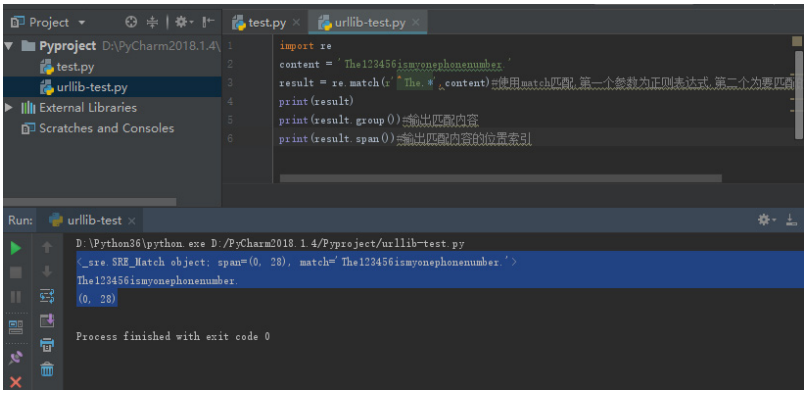

图 2-19　公用匹配用法

3. search()方法

与 match()方法不同，search()方法不需要从头部开始匹配，代码如下：

```
import re
content = 'OtherThe123456ismyonephonenumber.'
result = re.search('The.*?(\d+).*?number.',content)
print(result.group())
```

4. findall()方法

match()方法和 search()方法都是返回匹配到的第一个内容就结束匹配，findall()方法是返回全部符合匹配规则的内容，返回的是一个列表，代码如下：

```
import re
text = 'pyypppyyyypppp'
pattern = 'py'
for match in re.findall(pattern, text):
    print('Found {!r}'.format(match))
```

5. sub()方法

去除或替换匹配的字符。假如写 sub('\d+','-')，则把匹配的内容替换成'-'，例子如下：

```
import re
content='54abc59de335f7778888g'
content=re.sub('\d+','',content)
print(content)
```

2.4　实战案例：百度新闻的抓取

1. 获得 URL

打开百度，单击左上角的新闻，百度新闻如图 2-20 所示。

图 2-20　百度新闻

2. 元素审查

在打开的界面按下 F12 键或右击选择“检查”选项，选择所要抓取的新闻，能够看到的 URL 地址为：http://news.baidu.com/，百度新闻域名如图 2-21 所示。

图 2-21　百度新闻域名

3. 导入模块

导入 urllib 和 re 两个模块，代码如下

```
import urllib.request
import re
```

4. 请求 http 页面，响应状态

首先确认 URL 地址后向网站发送请求，查看响应状态码如图 2-22 所示。

```
import requests
url = "http://news.baidu.com"
data = requests.get(url)
print(data)
```

requests.get 发送地址，尝试获得 URL 地址的响应状态后输出响应状态码，返回的值为 200，该类型状态码表示动作被成功接收、理解和接受。

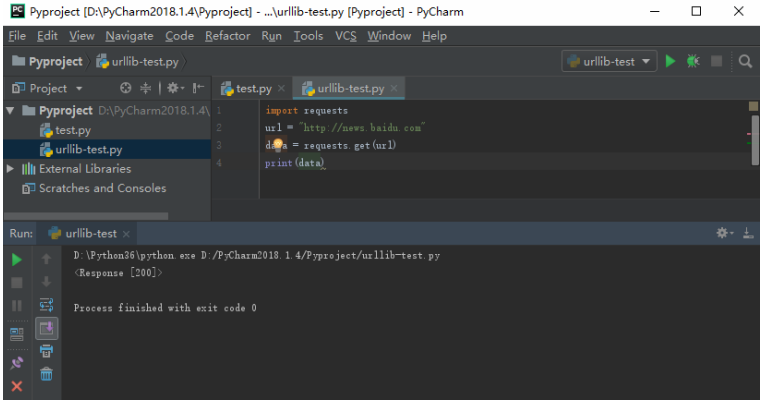

图 2-22　查看响应状态码

5. 抓取百度新闻

在进行模块与库的添加时，要注意 Python 库及模块是否已存在，模块分别是 re、urllib.request 及 datetime。

6. 寻找数据特征

打开百度新闻，网址 URL 为 http://news.baidu.com/，打开网页，按住 F12 键显示开发者工具，浏览器开发者工具如图 2-23 所示。

图 2-23　浏览器开发者工具

7. 查看 HTML 信息

需要抓取的是这个页面每一条新闻的标题，右击一条新闻的标题，选择“查看”选项，出现图 2-24 所示的窗口，则图片中红框的位置便是那一条新闻标题在 HTML 中的结构、位置和表现。

8. 分析网页源代码

从上一步骤中，经由对网页的源代码来进行分析，小标题都位于<a>的标签下面，这时候就可以使用正则的惰性匹配(.*?)来进行对标题的匹配，抓取标题的正则写法如图 2-25 所示。

图 2-24　查看网页新闻链接

```
pattern = re.compile('<li>\n<a href=".*?" mon=".*?" target=".*?">(.*?)<')
hotNews = re.findall(pattern,content)
```

图 2-25　抓取标题的正则写法

9. 准备抓取数据

在把准备工作完成后，开始抓取页面中的数据，整体代码展示如图 2-26 所示。

10. 抓取结果展示如图 2-27 所示。

```
import re
import datetime
import urllib.request
url = 'http://news.baidu.com'
content = urllib.request.urlopen(url).read().decode('utf-8')
with open('test.txt','w',encoding='utf-8') as f:
    f.write(content)

with open('test.txt','r',encoding='utf-8') as f:
    content = f.read()
pattern = re.compile('<li>\n<a href=".*?" mon=".*?" target=".*?">(.*?)<')
hotNews = re.findall(pattern,content)

for i in hotNews:
    print(i)
print(datetime.datetime.now())
```

图 2-26　整体代码展示

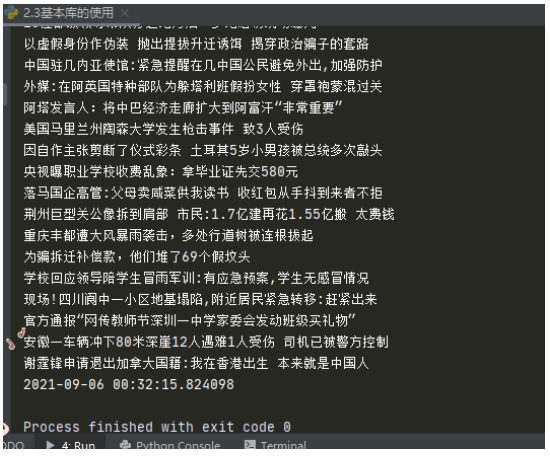

图 2-27　抓取结果展示

11. 设置时间点

在图 2-27 中能够看到抓取的内容，在最后一行显示着当前系统的时间，在实际环境中进行抓取的内容可能不止图 2-27 中的这一小部分，而数据内容过多会导致在处理数据时遇到一些问题，所以需要（或可以）在代码中添加一个时间戳。代码如下：

```
#使用到先前所添加的一个模块
import datetime
```

在代码的最后一行添加如下代码，这样在每次抓取完后，当前的系统时间会自动标记在后方。

```
print(datetine.datetime.now())
```

2.5　本章习题

一、单选题

1. Python 是一门（　　）语言。

A. 解释　　B. 编译

2. 以下选项中，Python 占位符和替换内容不对应的是（　　）。

A. %d 整数　　B. %f 浮点数　　C. %s 字符串　　D. %x 复数

3. 常规的 Python 格式化输出不包括（　　）。

A. []%　　B. .format()　　C. %s

4. Python list 本质上是一种（　　）数据结构。

A. 列表　　B. 线性表　　C. 栈　　D. 队列

5. list 和 tuple 数据结构最大的区别在于（　　）。

A. list 可以索引元素，tuple 不可以

B. tuple 可以索引元素，list 不可以

C. list 数据是不可变的，tuple 是可变的

D. list 数据是可变的，tuple 是不可变的

6. 以下选项中，（　　）不属于 Python 数据采集相关的库。

A. urllib　　B. requests　　C. lxml　　D. openpyxl

7. 以下选项中，Python 机器学习领域的第三方库是（　　）。

A. scipy　　B. PyTorch　　C. PyQt5　　D. requests

8. 以下选项中，不是 Python 关键字的是（　　）。

A. do　　B. return　　C. except　　D. while

二、判断题

1. 在函数内部没有任何声明的情况下直接为某个变量赋值，这个变量一定是函数内部的局部变量。（　　）

2. 定义类时如果实现了_contains_()方法，该类对象即可支持成员测试运算 in。（　　）

3. 定义类时如果实现了_len_()方法，该类对象即可支持内置函数 len()。（　　）

4. 定义类时如果实现了_eq_()方法，该类对象即可支持运算符==。（　　）

5. Python 常用的数据处理库包括 numpy、pandas 和 PIL。（　　）

6. 定义类时如果实现了_pow_()方法，该类对象即可支持运算符**。（　　）