**温州大学计算机与人工智能学院**

**Python应用开发 实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程名称：** | Python应用开发 | | | | |
| **班 级：** | 22网工2 | **姓名：** | 吕琳梅 | **学号：** | 22211835211 |
| **实验地点：** | 5B103 | **日期：** | 2024/12/19 | | |

# 程序简介

在移动互联网迅速发展的背景下，对APK文件的批量下载与分析对于移动应用生态的研究和安全评估具有重要意义。然而，许多应用商城为了保护数据接口，增加了获取APK资源的复杂性。

本系统是一个基于Python的自动化APK批量下载工具，主要针对应用宝（腾讯应用商店）进行数据采集和APK下载。系统采用模块化设计，分为三个主要模块：URL采集模块、应用详情提取模块和APK下载模块。系统能够自动获取应用商店排行榜前500个应用的详细信息，并进行批量下载。主要包括自动获取应用商店排行榜应用URL、提取应用详细信息并结构化存储、断点续传式批量下载APK文件以及MD5校验确保下载完整性等核心功能。

# 开发要点

## 2.1 要点1 反反爬取

为了绕过应用宝反爬机制，本程序采用以下策略提高爬虫的稳定性和成功率：

* **浏览器模拟：**使用Selenium WebDriver模拟真实浏览器行为，配置Chrome选项以模拟移动设备访问。
* **请求控制：**实现动态延迟机制，每次请求间隔3-5秒，添加合理的User-Agent和请求头。
* **显式等待：**利用 Selenium 的 WebDriverWait 确保页面动态加载完成后再进行抓取，防止因页面未完全渲染导致的数据提取失败。

## 2.2 要点2 数据爬取与信息提取

本程序通过以下步骤实现对APK信息的精确提取：

* **动态页面处理：**使用Selenium模拟页面滚动触发动态加载。
* **API分析与数据获取：**通过浏览器开发者工具分析应用商城的网络请求，找到获取APK相关信息的接口，利用 Selenium 获取页面中嵌入的动态数据。
* **数据完整性验证：**使用tqdm显示下载进度，实现分块下载和进度更新。

## **2.3要点3数据存储**

为保证数据的规范性和完整性，本程序对提取到的信息进行结构化存储

* **数据存储和去重：**将提取到的字段存储为结构化的CSV文件，包含以下列：应用名称、应用包名、标签、评分、开发者、下载地址和APK文件的MD5码。避免重复抓取相同的应用页面链接，程序使用一个集合 seen\_urls 来存储已抓取的链接。
* **APK下载与校验：**根据获取的下载地址，程序依次下载前500个APK文件，并计算每个APK文件的MD5码，将其添加到对应的CSV记录中。
* **文件处理：**采用追加模式写入，支持断点续传。

# 运行效果、运行关键步骤

该程序的主要目标是通过爬虫技术，从指定的应用商城中批量获取前500个APK的下载链接及相关信息，并将其保存为CSV文件，同时下载APK文件并计算其MD5码。本实验利用 Selenium 模拟移动设备用户行为，结合XPath定位技术，提取所需数据并存储。程序的整体流程分为三个主要阶段：**API分析与获取**、**数据爬取与信息提取**、**数据存储、APK下载与校验**。以下是每个步骤的详细实现过程：

## **3.1步骤1：API分析与获取**

### 识别API接口

许多网站的 API 接口是通过 JavaScript 动态加载的，要抓取这些 API 的数据，需要通过浏览器的开发者工具（Network 面板）找到相应的 API 请求。常见的 API 接口通常是通过 HTTP 请求（如 GET、POST）获取数据，返回 JSON 或 XML 格式的响应。

打开 Chrome 浏览器的开发者工具。在 Network 面板中，选择 XHR 或 Fetch 类型，这样可以过滤出所有的网络请求，尤其是 API 请求。执行页面上的操作（如点击按钮、滚动页面等），查看是否会发送新的网络请求。通常，API 请求会出现在 Network 面板中，以 XHR 或 Fetch 请求的形式出现。

点击某个 API 请求，查看它的详细信息，包括请求 URL、请求头、请求参数、响应内容等。

### **分组请求与响应**

**请求的组成：**

* **请求方法：** GET  
  在该接口中，GET 请求方法被用来请求 APK 文件。GET 请求是最常见的 API 请求方式，用于获取服务器上的资源。
* 请求 URL：

https://imtt2.dd.qq.com/sjy.00009/sjy.00004/16891/apk/B46EF5B29B2A4CDD8835A040CCCC3222.apk?fsname=com.tencent.mobileqq\_9.1.25.apk

该 URL 通过 HTTPS 协议请求一个 APK 文件，并附带查询参数 fsname，标识下载的 APK 文件名。

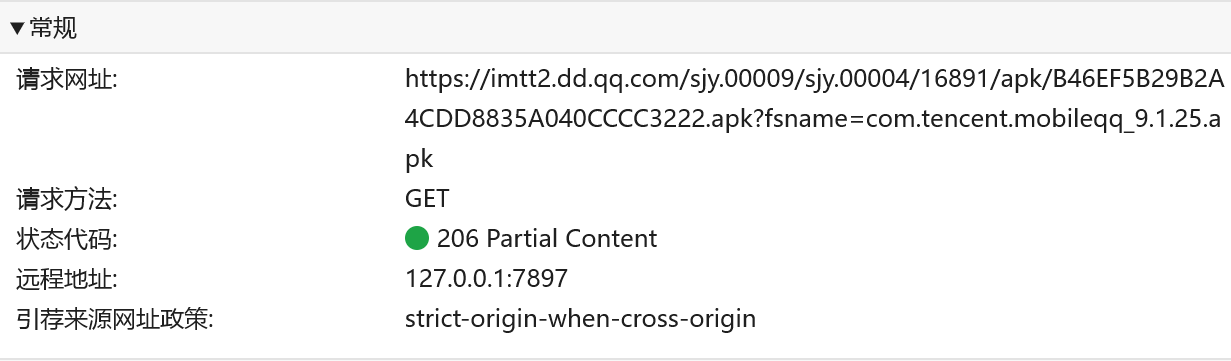


图 1 常规示例

* 请求头：

User-Agent模拟 Android 设备的浏览器行为（在此案例中，User-Agent 是针对 Android 设备设置的）。

Cookie包含了用户的会话信息，有助于在请求中保持用户的登录状态。

Range该请求通过 bytes 参数表示请求的是 APK 文件的部分内容，这样可以支持大文件的断点续传。

* 请求参数：

fsname: com.tencent.mobileqq\_9.1.25.apk该参数指定了 APK 文件的名称。

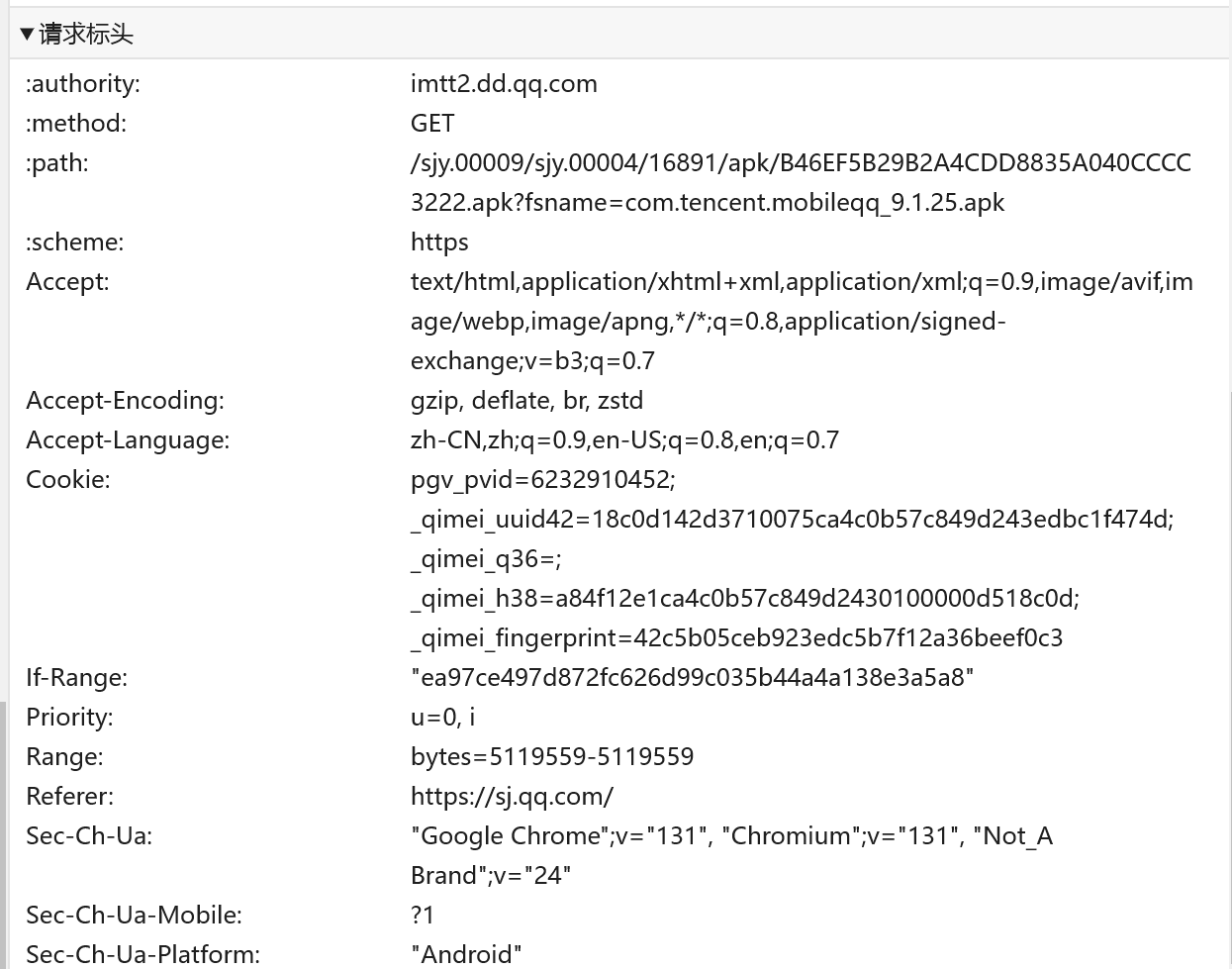


图 2 请求头示例1

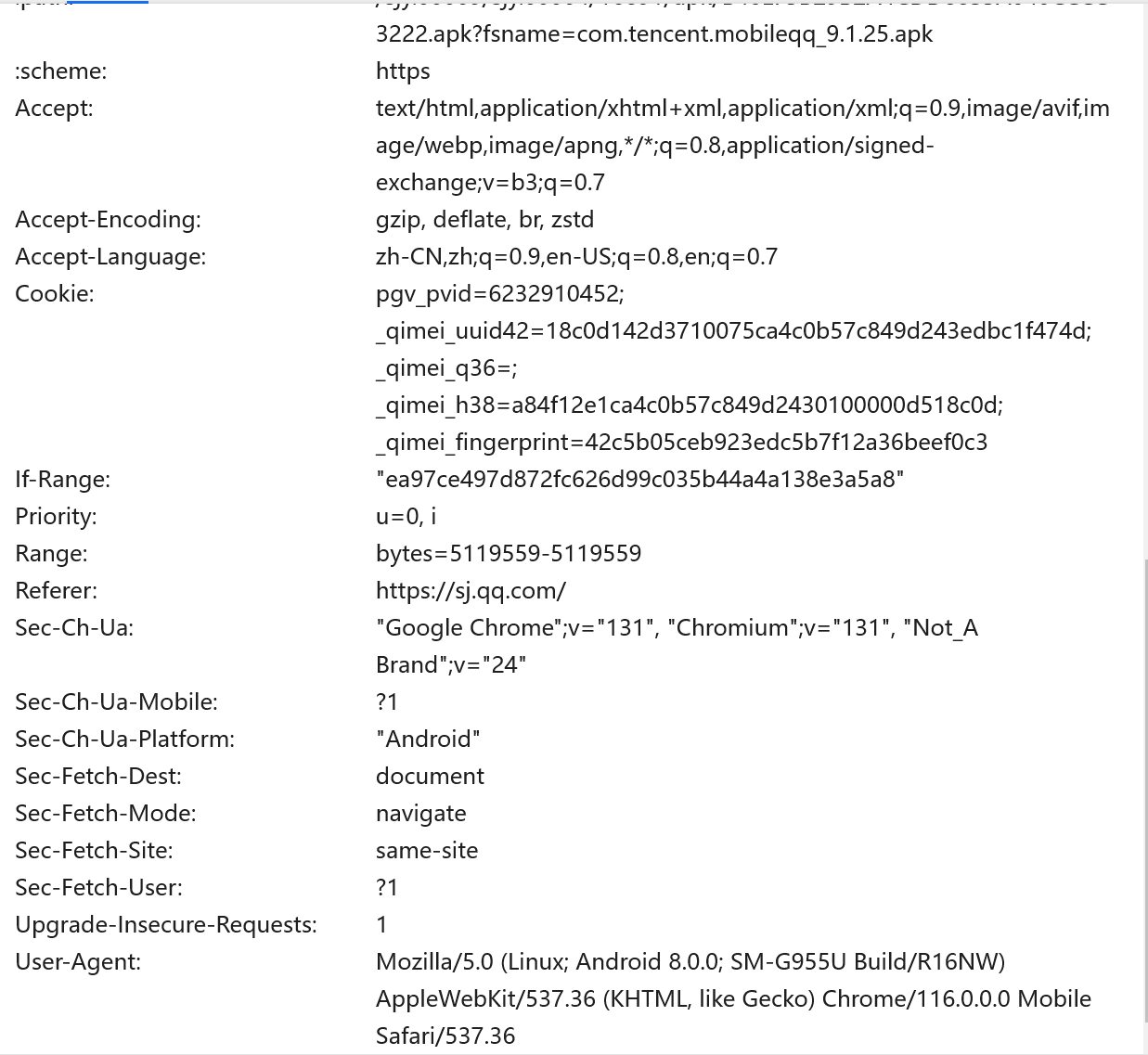


图 3 请求头示例2

**响应的组成：**

* **响应状态码：206 (Partial Content)**

状态码 206 表示服务器成功处理了请求，并返回部分文件内容。此类响应常见于大文件下载过程中，当用户请求文件的特定部分时，服务器会返回相应的字节范围。

* 响应头：

Content-Type: application/vnd.android.package-archive指定返回内容是 APK 文件，表明这是一个 Android 应用包。

Content-Range: bytes 5119559-5119559/322942059该响应头表示返回的是 APK 文件的一个字节范围，这通常用于大文件的下载，指示当前下载的数据片段。

* 响应体：响应体包含的是 APK 文件的实际内容，在请求成功后，客户端会根据响应体保存 APK 文件。

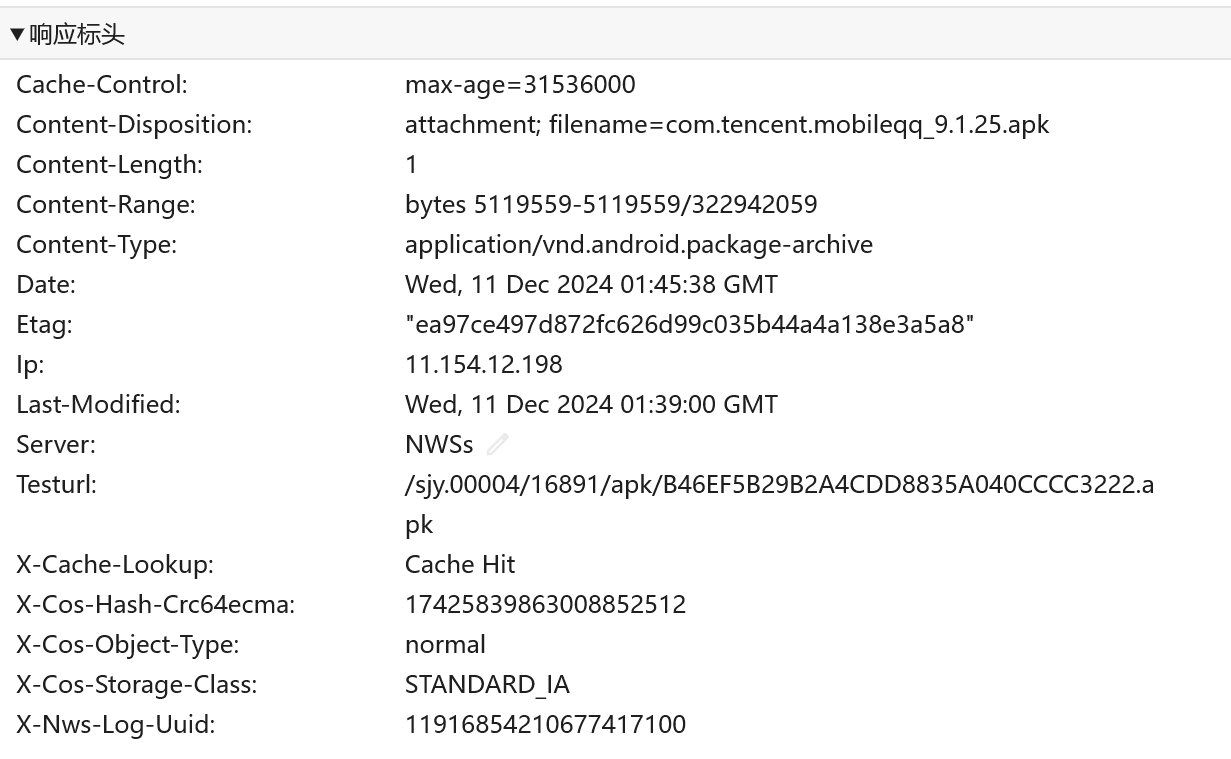


图 4 响应头示例

## **3.2 步骤2：数据爬取与信息提取**

### 3.2.1 数据爬取

数据爬取部分的主要任务是通过模拟用户的浏览和下滑行为，自动化获取腾讯应用宝上的应用程序详细信息，具体为所有应用程序的详情页面链接。实验中的数据爬取过程如下：

使用 webdriver.ChromeOptions() 配置 Chrome 浏览器的选项，随后通过 webdriver.Chrome() 启动浏览器实例。

程序首先加载腾讯应用宝网站首页（https://sj.qq.com/app），并等待页面加载完毕。通过 time.sleep(5) 给页面足够的时间来完成初始渲染。使用 Selenium 获取当前页面的 HTML 源代码，并使用 BeautifulSoup 对其进行解析。

通过 soup.find\_all() 方法，从页面源代码中找到所有符合条件的 <a> 标签，每个标签的 href 属性中包含了指向应用详细页面的相对链接，具体代码如图 5所示。

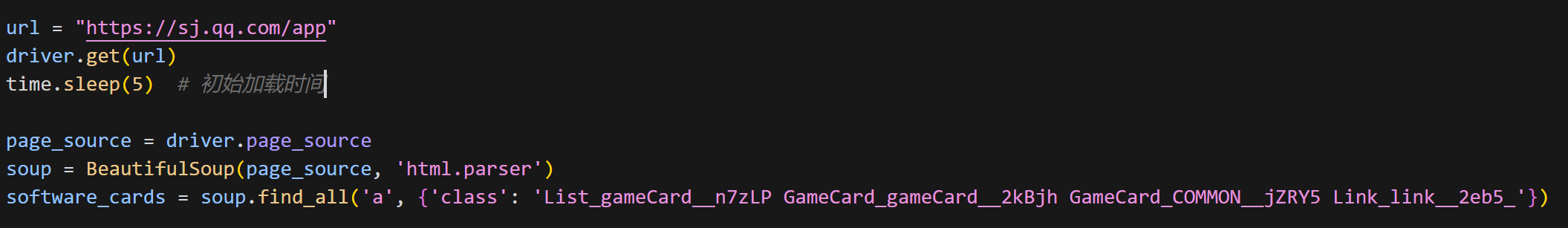


图 5 网页初始化

为避免重复抓取相同的应用页面链接，程序使用一个集合 seen\_urls 来存储已抓取的链接。每次抓取到新的链接时，程序都会检查该链接是否已存在于集合中，如果不存在，则将其添加到 software\_list 列表中。这样可以保证抓取到的链接既不重复又按照抓取的顺序保存在列表中，代码如下图 6所示。



图 6 App链接去重

为了获取更多的应用程序链接，程序使用滚动页面的方式加载更多的应用卡片。通过 JavaScript 执行 window.scrollTo() 方法，模拟用户向下滚动页面。每次页面滚动后，程序会等待页面重新加载并再次抓取页面源代码。为了避免请求过于频繁导致网站反爬虫机制的触发，每次滚动操作后，程序暂停 3 秒，以模拟真实用户的浏览行为。

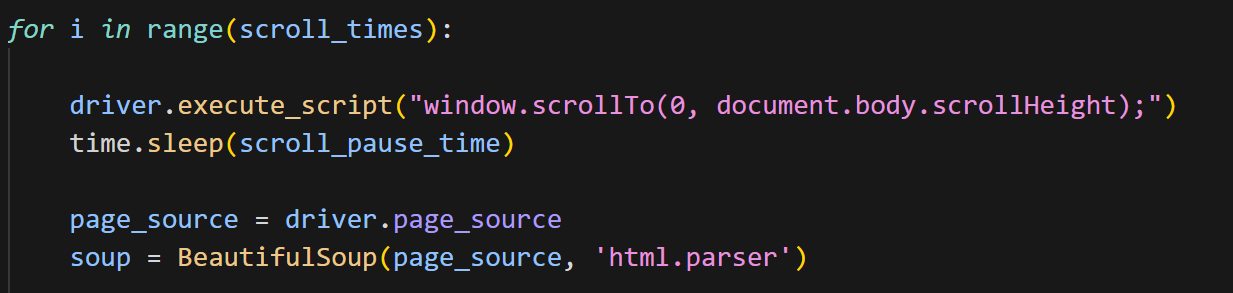


图 7 滚动翻页机制

如上图 7所示每次翻页后，程序再次使用 BeautifulSoup 提取页面中的应用卡片链接，确保覆盖到页面中所有已加载的内容。

所有爬取到的应用页面链接（去重后的链接）将保存在本地的 appdetail\_urls.txt 文件中，代码如下图 8所示，以便后续分析和处理。每个链接占据一行，确保文件格式简洁且易于读取，txt文件中的链接如下图 9所示。

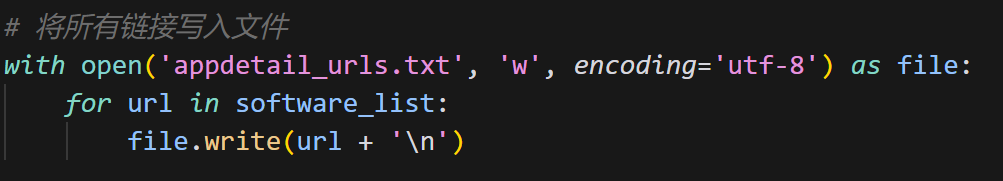


图 8 存储详情链接到txt文件

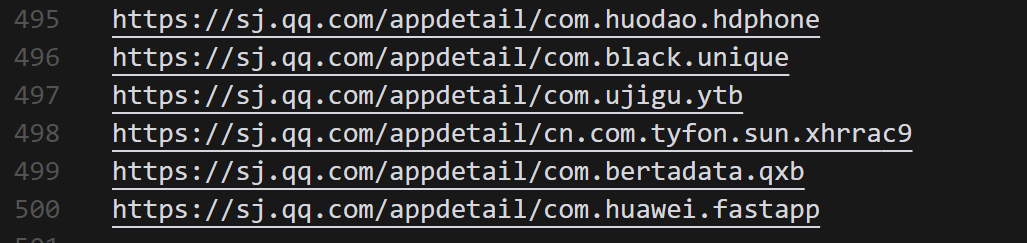


图 9 txt文件内容（app详情页链接）

### **3.2.2 信息提取**

在爬取每个APP的相关信息时，根据网页结构和相应的 XPath 路径定位需要的字段，本次信息提取示例网页如下图 10和图 11所示：



图 10 提取示例网页部分信息1



图 11提取示例网页部分信息2

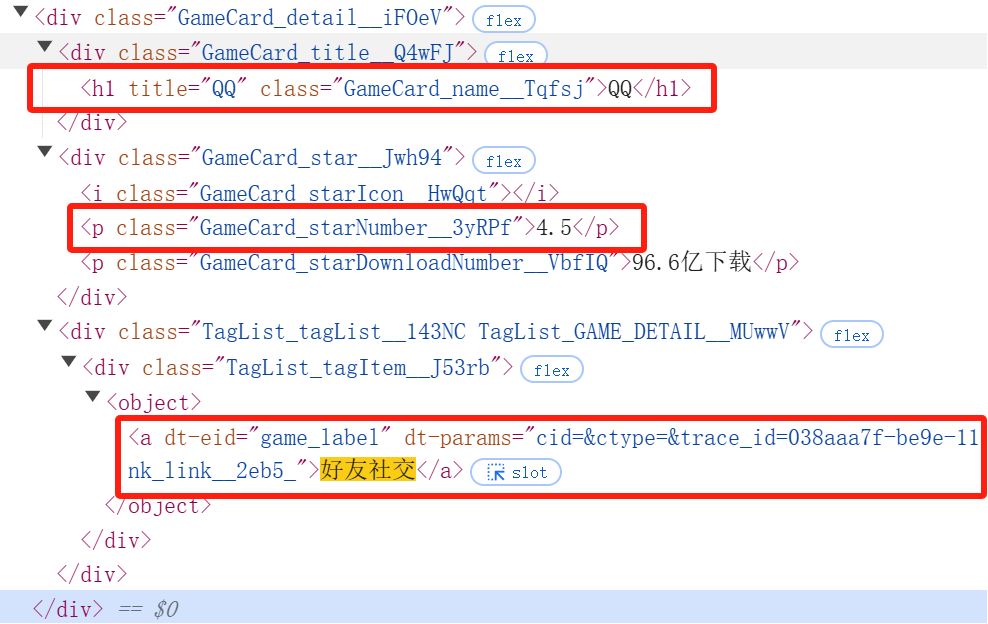


图 12 示例App名称、评分、标签提取



图 13 示例开发者名称提取

除上述信息以外，腾讯应用宝详情页的部分信息嵌套于动态脚本数据中。利用 JSON 解析技术提取 \_\_NEXT\_DATA\_\_ 脚本内的数据，以获取版本号与文件 MD5 值，基于提取到的 MD5 值，进一步构造下载链接，代码如下图 14所示。

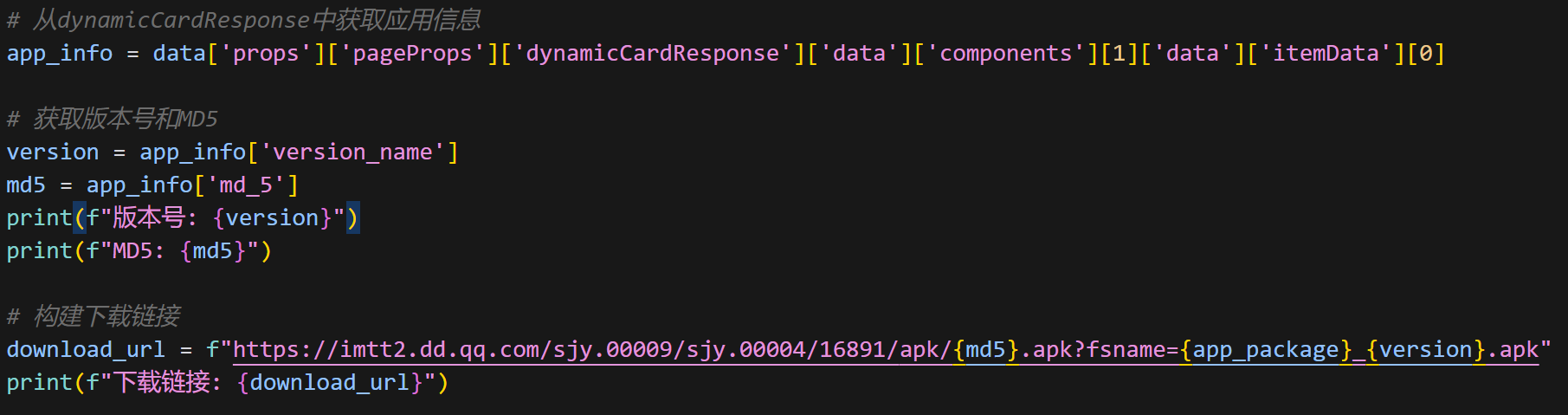


图 14 示例App版本号和MD5提取

以下是每个字段的 XPath 表达式：

表 1 App信息提取方式

|  |  |  |
| --- | --- | --- |
| **字段** | **示例** | **Xpath** |
| **Name** | QQ | soup.find('h1', class\_='GameCard\_name\_\_Tqfsj').text |
| **Package** | com.tencent.mobileqq | app\_package = url.split('/')[-1] |
| **Tags** | 好友社交 | soup.find('a', class\_='TagList\_tagName\_\_rrPrj TagList\_tagName-0\_\_1lsEk Link\_link\_\_2eb5\_').text |
| **Score** | 4.5 | soup.find('p', class\_='GameCard\_starNumber\_\_3yRPf').text |
| **Developers** | 深圳市腾讯计算机系统有限公司 | dev\_info = soup.find('div', class\_='AppInfo\_info\_\_p5Jlf')      dev\_p\_tags = dev\_info.find\_all('p') |
| **Version** | 9.1.25 | version = app\_info['version\_name'] |
| **MD5** | B46EF5B29B2A4CDD8835A040CCCC3222 | md5 = app\_info['md\_5'] |

其中app\_info = data['props']['pageProps']['dynamicCardResponse']['data']

['components'][1]['data']['itemData'][0]。

每个 XPath 路径的设计都是基于网页的结构，确保准确定位所需的信息。每次提取完一个字段后，程序会把它们存储在csv文件中。

## **3.3步骤3：数据存储**

本实验通过提供的代码成功抓取了某应用商店中各个应用的详细信息。对于每个应用链接，代码依次提取了其应用名称、包名、标签、评分、开发者信息、版本号、MD5 校验值及下载链接。爬取的数据存储在 CSV 文件中，csv文件格式如下表 2所示：

表2 Apk相关属性信息

|  |  |
| --- | --- |
| **列名** | **描述** |
| Name | 应用名称 |
| Package | 应用的包名 |
| Tags | 应用的标签 |
| Score | 应用评分 |
| Developers | 开发者名称 |
| Version | 应用版本号 |
| MD5 | APK 文件的 MD5 校验值 |
| Download\_URL | 应用的直接下载链接 |

为确保爬取到的应用数据能够正确存储，程序在存储数据之前，程序会检查目标 CSV 文件是否已经存在。如果文件不存在，则创建文件并写入表头。如果文件存在，则以追加模式写入新数据。爬取的数据首先被存储为字典，并转换为 Pandas 数据框（DataFrame），如下图 15所示为模块的代码实现：



图 15 存储模块文件代码实现

程序运行后，所有爬取到的应用详情信息成功存储到 app\_details.csv 文件中。文件内容如下图 16所示：

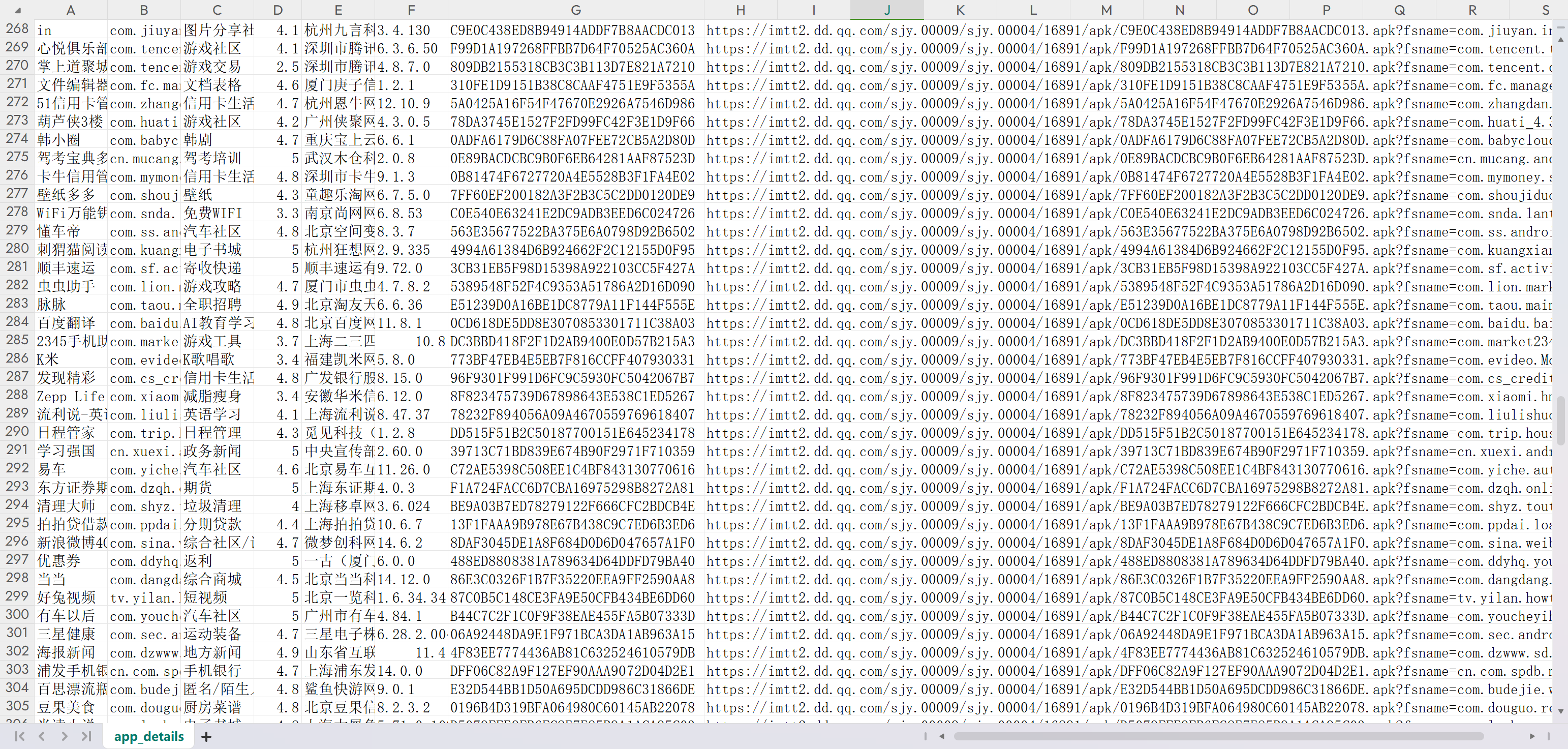


图 16 存储App信息文件内容

## **3.4步骤4：APK下载与校验**

通过加载并保存每次下载的状态，我们能够追踪哪些应用已经成功下载，哪些未能成功下载，并且能够在系统重启后从上次中断的地方继续下载。

APK下载功能由 download\_apk 函数实现，该函数负责通过提供的下载链接获取APK文件并将其保存至本地指定目录。具体代码如下图 17所示：

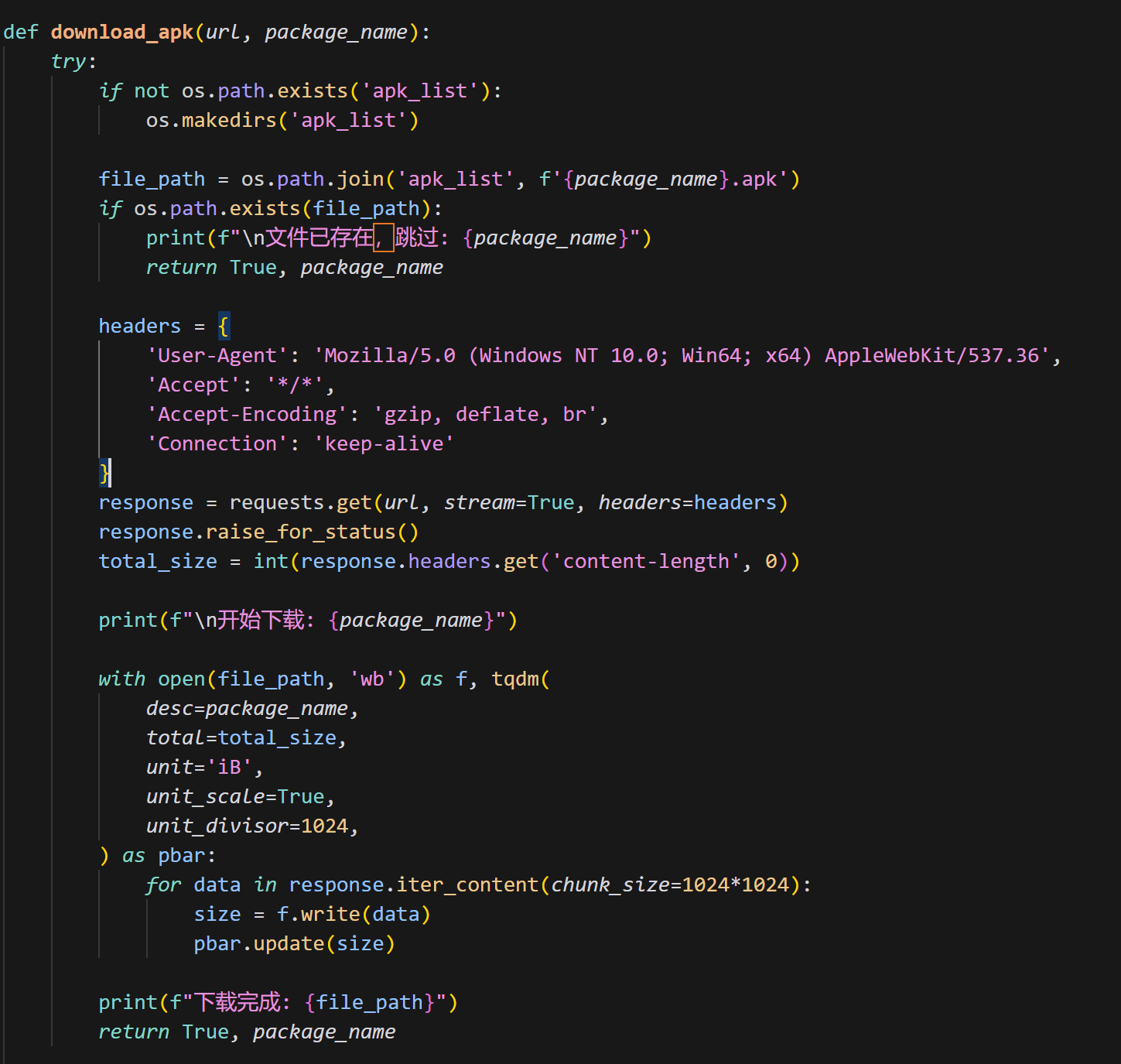


图 17 Apk下载

首先检查是否存在存储APK文件的文件夹（apk\_list），若不存在则自动创建该文件夹，以确保文件能够正确保存。下载前，程序会先判断是否已下载过该文件如果该文件已经存在，则跳过下载，避免重复下载。

下载过程中使用 tqdm 库显示下载进度条，展示文件下载的进度和剩余时间。下载的数据通过二进制写入文件，直到所有数据下载完成并保存，如下图 18所示。

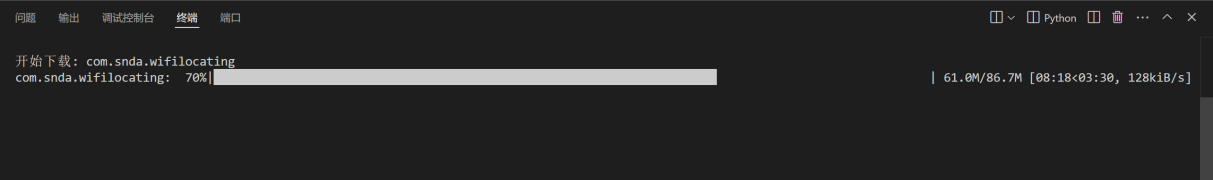


图 18 下载Apk文件

下载APK文件后，为确保文件的完整性和正确性，本实验实现了对下载文件的MD5校验。在下载过程中，程序会对每个APK文件进行MD5校验，并与文件的预期MD5值进行比较。若校验值不匹配，则表示文件下载有误，最终下载完成，结果如下图 19所示。

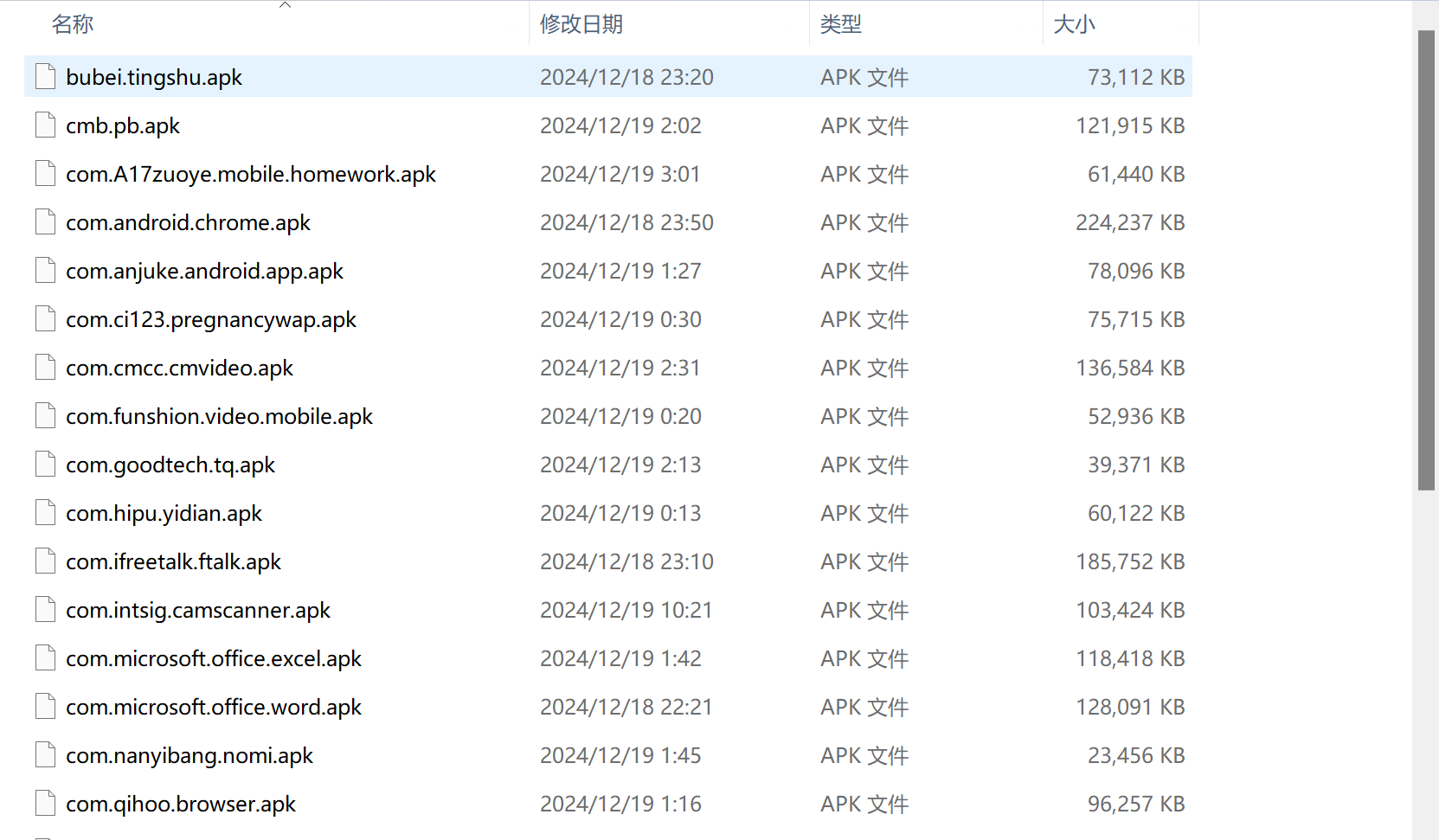


图 19 apk文件包下载完成