****

**软 件 学 院**

**实践环节报告**

**课题名称： 软件工程实践一**

**专业： 软件工程**

**班级： 卓越161**

**学号： 201619140220**

**学生姓名： 崔烁豪**

**指导老师： 韩玉民**

2018年9月21日

**中原工学院软件学院**

**软件工程实践一课题任务书**

|  |  |  |  |
| --- | --- | --- | --- |
| **姓 名** | 崔烁豪 | **班级** | 卓越161班 |
| **题 目** | 研信网研发−数据分析与获取 | | |
| **设**  **计**  **任**  **务** | 为了帮助选择考研的同学能在选择考研院校时，能对比较适合自己的院校的历年初试复试有一个系统的了解，为考研学生提供信息支持，研发一个基于Web的考研院息分析网−研信网。主要功能包括：  1.院校信息展示  主要包括院校名称、所在地、院校隶属、院校特征、有无研究生院、是否自划线等。  2.招生简章展示  提供该院校公布的最新招生计划等内容  3.专业目录信息展示  主要包括院系所，专业，研究方向，学习方式（全日制/非全日制），拟招生人数，考试范围，历年分数线，学制等。  4.考试范围  考试范围包括以下属性：年份，考试科目，单科分数要求等。  5.国家线管理  展示各年份每个院校的不同专业的国家复试线，包括单科线。  6.学校历年复试线与录取情况  主要包括：年份、分数、报考人数、录取人数等。  数据存储要求：采用阿里云服务器存储数据。  开发工具：MySQL5.6，MyEclipse，PyCharm | | |
| **时**  **间**  **进**  **度** | 第1周（2018-09-03~2018-09-09）：对需求进行深入分析，并设计出架构与数据库，仔细理解需求，规划功能。  第2周（2018-09-10~2018-09-16）：学习并完成每一步的开发，在过程中发现问题并改进问题，学习使用第三方的API来完成系统，并进行下一步的功能该进，并优化界面。  第3周（2018-09-17~2018-09-23）：调试测试，通过实际用户的操作，提出建议并修改，撰写报告。 | | |
| **原 主始 要资 参料 考与 文 献** | [01] 石志国，薛为民，董洁.JSP应用教程.北京：清华大学出版社.北京交通大学出版社，2004  [02] 李曙光.JSP开发实例完全剖析[M].北京：中国电力出版社，2006  [03] 杨易.JSP高级程序设计[M].人民邮电出版社，2006  [04] Vivek Chopra.JSP高级程序设计，机械工业出版社，2001  [05] 申吉红，廖学峰，余健.JSP课程设计案例精编[M].北京：清华大学出版社，2007  [06] 卢潇.软件工程[M].北京：清华大学出版社；北京交通大学出版社，2005  [07] 黑马程序员.Java EE企业级应用开发教程.北京：人民邮电出版社，2017 | | |

# 摘要

有研究调查表明，有七成以上的在校大学生有明确的考研意愿。调查结果显示，75.3%的同学有较为明确的考研意向，仍在考虑的占20.2%。而根据参与调查的学生的各种维度进行分类发现，这种近乎盲目的考研热几乎是不分年级、不分性别、不分学校层次的。而考研是为了什么呢？提高就业竞争力，给自己的学历加码是主要原因。大多数考研人害怕就业受歧视，非全日制研究生接受率不高；考研择校更关注学校整体知名度而非学术水平。因此，我们为了方便考研的同学，把考研的主要信息整合到一个网站，同学们可以很容易的获取自己想了解的内容。

研信网，为提高大学生选择考研目标和考研专业的效率而生，更方便、更系统的查看考研院校和专业信息。不用再一个一个的去学校官网查看比对了。而且，本系统的另一个重要特点是界面友好、易于操作，有很好的可视化效果。

本系统采用JAVA、Servlet、JSP、Jfinal框架开发后台，在阿里云服务器上部署MySQL数据库，用JavaScript，HTML，CSS，BootStrap框架开发前端，用Python、XPATH抓取院校专业等数据。

本文共5章，第1章主要介绍了研信网开发的背景，研究意义及系统功能，第2章是对研信网的系统分析，第3章是研信网的数据库详细设计，第4章是对研信网的具体实现进行阐述，第5章介绍在实践成果完成过程中的一些心得体会，以及对此次实践成果的评价，进行总结。

关键字：考研；信息；网站；数据获取

至少3个关键字，要能反映题目

# 目录

[摘要 III](#_Toc528686043)

[目录 1](#_Toc528686044)

[第1章 概述 1](#_Toc528686045)

[1.1 课题背景 1](#_Toc528686046)

[1.2 研信网的功能 1](#_Toc528686047)

[1.3 研信网的开发意义 2](#_Toc528686048)

[1.4 成员与分工 2](#_Toc528686049)

[第2章 系统分析 3](#_Toc528686050)

[2.1技术分析 3](#_Toc528686051)

[2.2需求分析 3](#_Toc528686052)

[第3章 数据库分析与设计 5](#_Toc528686053)

[3.1 实体分析 5](#_Toc528686054)

[3.2关系设计 8](#_Toc528686055)

[3.3数据表设计 9](#_Toc528686056)

[第4章 系统实现 12](#_Toc528686057)

[4.1数据分析与获取项目流程 12](#_Toc528686058)

[4.2功能实现 14](#_Toc528686059)

[4.2.1院校信息分析与获取 14](#_Toc528686060)

[4.2.2院校招生简章分析与获取 15](#_Toc528686061)

[4.2.3院校专业信息分析与获取 16](#_Toc528686062)

[4.2.4院校专业研究方向信息分析与获取 18](#_Toc528686063)

[4.2.5院校专业考试范围信息分析与获取 19](#_Toc528686064)

[4.2.6院校专业分数线信息分析与获取 21](#_Toc528686065)

[4.3数据获取结果 22](#_Toc528686066)

[4.4系统的发布 23](#_Toc528686067)

[第5章 结束语 24](#_Toc528686068)

[指导老师评语 26](#_Toc528686069)

[附录 27](#_Toc528686070)

[1.爬取院校信息的主要代码： 27](#_Toc528686071)

[2.把院校信息数据插入院校表的主要代码： 28](#_Toc528686072)

[3.爬取招生简章的主要代码： 29](#_Toc528686073)

[4.把招生简章数据插入招生简章表的主要代码： 31](#_Toc528686074)

[5.爬取专业目录的主要代码： 33](#_Toc528686075)

[6.把专业目录数据插入专业目录表的主要代码： 34](#_Toc528686076)

[7.爬取专业研究方向的主要代码： 36](#_Toc528686077)

[8.把专业研究方向数据插入研究方向表的主要代码： 39](#_Toc528686078)

[9.爬取专业考试范围的主要代码： 41](#_Toc528686079)

[10.把专业考试范围数据插入考试范围表的主要代码： 44](#_Toc528686080)

[11.爬取专业分数线的主要代码： 46](#_Toc528686081)

[12.把专业分数线数据插入分数线表的主要代码： 49](#_Toc528686082)

# 第1章 概述

本章主要阐述研信网研发的开发背景、网站功能及开发意义。

## 1.1 课题背景

有研究调查表明，有七成以上的在校大学生有明确的考研意愿。调查结果显示，75.3%的同学有较为明确的考研意向，仍在考虑的占20.2%。而根据参与调查的学生的各种维度进行分类发现，这种近乎盲目的考研热几乎是不分年级、不分性别、不分学校层次的。而考研是为了什么呢？提高就业竞争力，给自己的学历加码是主要原因。大多数考研人害怕就业受歧视，非全日制研究生接受率不高；考研择校更关注学校整体知名度而非学术水平。因此，我们为了方便考研的同学，把考研的主要信息整合到一个网站，同学们可以很容易的获取自己想了解的内容。

## 1.2 研信网的功能

研信网主要任务就是考研信息的检索与展示，信息的来源主要有研招网、文都考研网等，信息获取的方式是利用python网络爬虫的方式进行获取的。主要针对高校考研学生，帮助他们更加方便的获取院校信息，确定自己的目标院校。

它主要包含以下几个方面：

1.信息获取

使用python爬取研招网、文都考研网等，获取院校相关信息

2.信息展示

将获取的信息通过网页进行展示

3.信息检索

因为信息量比较大，提供信息检索功能帮助同学们更方便的获取想要信息

## 1.3 研信网的开发意义

对于准备考研的学生们来说，院校选择的重要性不言而喻，要想选择院校，就需要对该院校的相关考研信息有一定的了解。研信网整合了各大考研网的院校关键信息，可以帮助同学们更加方便快速的获取关键信息，从而确定目标院校

## 1.4 成员与分工

表1.1 成员与分工表

|  |  |
| --- | --- |
| **分工**  **成员** | **任务** |
| 季波 | 后台开发 |
| 葛文杰 | 前端设计 |
| 崔烁豪 | 数据获取 |

# 第2章 系统分析

本章主要是对研信网进行合理性的分析，通过技术分析、需求分析这两个方面连描述这个项目。

## 2.1技术分析

研信网的研发过程中，我们小组三个人进行了合理的分工，葛文杰负责前端UI设计，季波负责后台开发，我负责数据获取。

前端UI设计用到的技术有：JavaScript，HTML，CSS，BootStrap框架，用到的开发工具有：Hbuilder等。

后台开发用到的技术有：JAVA、Servlet、JSP、Jfinal框架，用到的开发工具有：MyEclipse、Tomcat等。

数据获取用到的技术有：Python、XPATH、JS2XML，用到的开发工具有：PyCharm、Google Chrome等。

除此之外、我们的数据库用的是MySQL，部署在阿里云服务器上。

## 2.2需求分析

研信网所需实现的功能模块图如图2.1所示.



图2.1 系统功能模块图

用户需求如下：

1.信息展示

首页主要展示院校的基本信息，包括名称、特性、所在地、隶属、是否有研究生院、是否是自划线院校，点击院校名称跳转至院校详情页面

院校详情页面分为四个目录-招生简章、专业目录、分数线、报录信息，各个目录展示想对应的信息

2.信息检索

首页提供地区检索、特性检索、模糊检索的功能

# 第3章 数据库分析与设计

数据库拥有很强的功能，可以建立一个数据库把我们所需要的数据给存储起来，并可以进行增删改查的操作，还可以对数据库进行导入导出等操作。我们采用MySQL数据库来保存运动会管理系统的各项信息。

## 3.1 实体分析

研招网涉及的实体包括院校、招生简章、专业、研究方向、考试范围、分数线等。经过分析，院校实体E-R图如图3.1所示，招生简章实体E-R图如图3.2所示，专业实体E-R图如图3.3所示，研究方向实体E-R图如图3.4所示，考试范围实体E-R图如图3.5所示，分数线实体E-R图如图3.6所示。



图3.1 院校实体E-R图



图3.2 招生简章实体E-R图



图3.3 专业实体E-R图



图3.4 研究方向实体E-R图



图3.5 考试范围实体E-R图



图3.6 分数线实体E-R图

## 3.2关系设计

这里有六个实体，分别是：院校、招生简章、专业目录、研究方向、考试范围、分数线。它们之间的关系是：招生简章属于院校，一对一。专业属于院校，多对一。研究方向属于专业，多对一。考试范围属于专业，一对一。分数线属于专业，多对一。研信网的系统E-R图如图3.7所示。



图3.7 系统总体E-R图

## 3.3数据表设计

研信网的数据库设计总共有六个表：院校表、招生简章表、专业目录表、研究方向表、考试范围表、分数线表。院校表如表3.1所示，招生简章表如表3.2所示，专业目录表如表3.3所示，研究方向表如表3.4所示，考试范围表如表3.5所示，分数线表如表3.6所示。

1.院校表，包含了所有考研院校名称、院校代码等信息。

表3.1 university数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| uni\_id | int | 11 | 否 | 是 |
| uni\_code | varchar | 10 | 是 | 否 |
| uni\_name | varchar | 20 | 是 | 否 |
| uni\_location | varchar | 10 | 是 | 否 |
| uni\_subjection | varchar | 10 | 是 | 否 |
| uni\_characteristic | varchar | 10 | 是 | 否 |
| uni\_graduate\_school | bool |  | 是 | 否 |
| uni\_self\_line | bool |  | 是 | 否 |

2.招生简章表，包含了每个院校对应的招生简章信息。

表3.2 recruit\_page数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| rec\_id | int | 11 | 否 | 是 |
| rec\_year | varchar | 5 | 是 | 否 |
| rec\_content | text |  | 是 | 否 |
| rec\_uni\_name | varchar | 20 | 是 | 否 |

3.专业目录表，包含了各个院校的专业信息。

表3.3 major\_catalog数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| maj\_id | int | 11 | 否 | 是 |
| maj\_year | varchar | 5 | 是 | 否 |
| maj\_code | varchar | 10 | 是 | 否 |
| maj\_name | varchar | 20 | 是 | 否 |
| maj\_college\_name | varchar | 20 | 是 | 否 |
| maj\_uni\_name | varchar | 20 | 是 | 否 |
| maj\_academic | varchar | 5 | 是 | 否 |
| maj\_system | int | 11 | 是 | 否 |
| maj\_transdisciplinary | bool |  | 是 | 否 |

4.研究方向表，包含了各个院校不同专业的研究方向信息。

表3.4 research\_direction数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| res\_id | int | 11 | 否 | 是 |
| res\_year | varchar | 5 | 是 | 否 |
| res\_name | varchar | 20 | 是 | 否 |
| res\_maj\_name | varchar | 20 | 是 | 否 |
| res\_college\_name | varchar | 20 | 是 | 否 |
| res\_uni\_name | varchar | 20 | 是 | 否 |
| res\_way | varchar | 10 | 是 | 否 |
| res\_register\_num | int | 11 | 是 | 否 |
| res\_enroll\_num | int | 11 | 是 | 否 |
| res\_recommend\_num | int | 11 | 是 | 否 |

5.考试范围表，包含了各个院校不同专业的初试考试范围信息。

表3.5 exam\_area数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| exa\_id | int | 11 | 否 | 是 |
| exa\_year | varchar | 5 | 是 | 否 |
| exa\_maj\_name | varchar | 20 | 是 | 否 |
| exa\_college\_name | varchar | 20 | 是 | 否 |
| exa\_uni\_name | varchar | 20 | 是 | 否 |
| exa\_subject1 | text |  | 是 | 否 |
| exa\_subject2 | varchar | 50 | 是 | 否 |
| exa\_subject3 | varchar | 50 | 是 | 否 |
| exa\_subject4 | varchar | 50 | 是 | 否 |

6.分数线表，包含了各个院校不同专业的初试分数线信息。

表3.6 score\_level数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| sco\_id | int | 11 | 否 | 是 |
| sco\_year | varchar | 5 | 是 | 否 |
| sco\_maj\_name | varchar | 20 | 是 | 否 |
| sco\_college\_name | varchar | 20 | 是 | 否 |
| sco\_uni\_name | varchar | 20 | 是 | 否 |
| sco\_country\_line | int | 11 | 是 | 否 |
| sco\_self\_line | int | 11 | 是 | 否 |
| sco\_reexamine\_line | int | 11 | 是 | 否 |
| sco\_subject1 | int | 11 | 是 | 否 |
| sco\_subject2 | int | 11 | 是 | 否 |
| sco\_subject3 | int | 11 | 是 | 否 |
| sco\_subject4 | int | 11 | 是 | 否 |

# 第4章 系统实现

软件设计是一个把软件需求变换成软件表示的过程。最初这种表示知识描绘出软件的总的框架,然后进一步细化，把它加工成在程序细节上非常接近于源程序的软件表示。从工程管理的角度来看，软件设计分两步完成。首先做概要设计，将软件需求转化成数据结构和软件的系统结构。然后是详细设计，即过程设计。通过对结构表示进行细化，得到软件的详细的数据结构和算法。

研信网设计阶段主要目的是将系统分析阶段所提出的反映了用户信息需求的系统逻辑方案转换成可以实施的基于计算机与通信系统的物理(技术)方案。

## 4.1数据分析与获取项目流程

数据分析与获取项目流程如图4.1所示。



图4.1 数据分析与获取项目流程

## 4.2功能实现

在研信网的设计开发过程中，准确明了的数据是必不可少的。数据在研信网中起着非常重要的作用。它的准确性直接影响到系统的使用性。研信网的数据获取实现过程如下所示。

### 4.2.1院校信息分析与获取

院校信息数据来源是中国研究生招生信息网，院校信息的数据来源网页如图4.2所示。可以看到，我们所要爬取的院校信息数据全部在这个网页，接下来就是用Python爬取这个网页上我们所需的数据，过程如下：

分析这个网页的URL，在Python中通过request请求获取这个URL下网页的所有数据，然后通过etree解析网页数据，再用XPATH从etree解析后的数据中分离出我们要的院校信息数据，并且调用已经写好的数据库操作模块，将数据存入数据库的院校表。



图4.2 院校信息数据来源网页

院校信息包括院校代码、院校名称、所在地、院校隶属、院校隶属、有无研究生院、是否自划线等。院校信息爬取成功后的展示页面如图4.3所示。



图4.3 院校信息页面

### 4.2.2院校招生简章分析与获取

招生简章数据来源是文都院校库网站，招生简章的数据来源网页如图4.4所示。可以看到，我们所要爬取的招生简章数据全部在这个网页，接下来就是用Python爬取这个网页上我们所需的数据，过程如下：

招生简章的数据在文都院校库中的院校详情下。根据文都院校库的首页链接，获取每个院校的详情链接，即招生简章数据所在网页，分析这个网页的URL，在Python中通过request请求获取这个URL下网页的所有数据，然后通过etree解析网页数据，再用XPATH从etree解析后的数据中分离出我们要的招生简章数据，并且调用已经写好的数据库操作模块，将数据存入数据库的招生简章表。



图4.4 招生简章数据来源网页

院校信息爬取成功后，要爬取院校的招生简章。一般情况下，一个院校对应一个招生简章，每个招生简章的所属院校名称都不能重复。根据实际情况爬取招生简章信息，招生简章爬取成功后，会自动存入数据库，并自动生成一个与招生简章唯一对应的ID。招生简章信息页面如图4.5所示。



图4.5 招生简章信息页面

### 4.2.3院校专业信息分析与获取

专业目录数据来源是文都院校库网站，专业目录的数据来源网页如图4.6所示。可以看到，我们所要爬取的专业数据全部在这个网页，接下来就是用Python爬取这个网页上我们所需的数据，过程如下：

专业的数据在文都院校库中的“院校”的“查看专业”下。根据文都院校库的首页链接，获取每个院校的“查看专业”链接，即专业目录数据所在网页，分析这个网页的URL，在Python中通过request请求获取这个URL下网页的所有数据，然后通过etree解析网页数据，再用XPATH从etree解析后的数据中分离我们要的专业目录数据。经过分析，我们发现，我们要的专业目录数据在table标签内，但是table标签内的数据经过etree解析后不显示。后来经过反复探索和分析，我们发现，我们要的专业目录数据在etree解析后script标签内，而且需要用js2xml取出来再解析才得到我们要的专业目录数据。然后，找到我们要的专业目录数据后，调用已经写好的数据库操作模块，将数据存入数据库的专业目录表。



图4.6 专业目录数据来源网页

院校招生简章信息爬取成功后，要爬取院校的专业信息。一般情况下，一个院校对应多个专业信息。根据实际情况爬取院校专业信息，院校专业信息爬取成功后，会自动存入数据库，并自动生成一个与院校专业信息唯一对应的ID。院校专业信息页面如图4.7所示。



图4.7 院校专业信息页面

### 4.2.4院校专业研究方向信息分析与获取

专业研究方向数据来源是文都院校库网站，专业研究方向的数据来源网页如图4.8所示。可以看到，我们所要爬取的研究方向全部在这个网页，接下来就是用Python爬取这个网页上我们所需的数据，过程如下：

专业研究方向数据在专业目录数据的“查看详情”下。根据文都院校库的首页链接，获取每个院校的“查看专业”链接，即专业目录数据所在网页，分析这个网页的URL，在Python中通过request请求获取这个URL下网页的所有数据，然后通过etree解析网页数据。我们要的专业目录数据在etree解析后script标签内，用js2xml取出来再解析后得到我们要的专业“查看详情”链接的重要片段，然后通过字符串拼接的方法得到我们要的专业“查看详情”链接。然后，在Python中通过request请求获取这个URL下网页的所有数据，然后通过etree解析网页数据，再用XPATH从etree解析后的数据中分离我们要的专业研究方向数据。最后，找到我们要的专业研究方向数据后，调用已经写好的数据库操作模块，将数据存入数据库的研究方向表。



图4.8 专业研究方向数据来源网页

院校专业信息爬取成功后，要爬取院校专业的研究方向。一般情况下，一个院校专业对应多个研究方向。根据实际情况爬取院校专业的研究方向信息，研究方向爬取成功后，会自动存入数据库，并自动生成一个与研究方向唯一对应的ID。研究方向信息页面如图4.9所示。



图4.9 研究方向信息页面

### 4.2.5院校专业考试范围信息分析与获取

专业考试范围数据来源是文都院校库网站，专业考试范围的数据来源网页如图4.10所示。可以看到，我们所要爬取的考试范围全部在这个网页，接下来就是用Python爬取这个网页上我们所需的数据，过程如下：

专业考试范围数据与专业研究方向数据在同一个网页。获取专业考试范围所在网页的URL之后，在Python中通过request请求获取这个URL下网页的所有数据，然后通过etree解析网页数据，再用XPATH从etree解析后的数据中分离我们要的专业考试范围数据。最后，找到我们要的专业考试范围数据后，调用已经写好的数据库操作模块，将数据存入数据库的考试范围表。



图 4.10 专业考试范围数据来源网页

院校专业信息爬取成功后，要爬取院校的考试范围。一般情况下，一个院校专业对应一个考试范围，每个考试范围的所属专业名称都不能重复。根据实际情况爬取考试范围信息，考试范围爬取成功后，会自动存入数据库，并自动生成一个与考试范围唯一对应的ID。考试范围信息页面如图4.11所示。

图4.11 考试范围信息页面

### 4.2.6院校专业分数线信息分析与获取

专业分数线数据来源是文都院校库网站，专业分数线的数据来源网页如图4.12所示。可以看到，我们所要爬取的分数线全部在这个网页，接下来就是用Python爬取这个网页上我们所需的数据，过程如下：

专业分数线数据与专业研究方向数据在同一个网页。获取专业考试范围所在网页的URL之后，在Python中通过request请求获取这个URL下网页的所有数据，然后通过etree解析网页数据，再用XPATH从etree解析后的数据中分离我们要的专业分数线数据。最后，找到我们要的专业分数线数据后，调用已经写好的数据库操作模块，将数据存入数据库的分数线表。



图 4.12 专业分数线数据来源网

院校专业信息信息爬取成功后，要爬取院校专业的分数线信息。一般情况下，一个院校专业对应一个分数线信息，每个分数线信息的所属专业名称都不能重复。根据实际情况爬取分数线信息，分数线信息爬取成功后，会自动存入数据库，并自动生成一个与分数线唯一对应的ID。分数线信息页面如图4.13所示。

图4.13 分数线信息页面

## 4.3数据获取结果

系统获取的数据库表一共有六个。每个表中有大量数据，每个表中的数据的爬取所用时间都很长。

六个表如下：

（1）院校表(University)

（2）招生简章表(recruit\_page)

（3）专业目录表(major\_catalog)

（4）研究方向表(research\_direction)

（5）考试范围表(exam\_area)

（6）分数线表(score\_level)

六个表的数据量如下：

（1）院校表(University)有853条数据

（2）招生简章表(recruit\_page)有955条数据

（3）专业目录表(major\_catalog)有42315条数据

（4）研究方向表(research\_direction)有38716条数据

（5）考试范围表(exam\_area)有37293条数据

（6）分数线表(score\_level)有40728条数据

因为爬取数据所用的时间受电脑性能和网速等不可控条件的影响，六个表爬取时所用的具体时间已无法详细提供。院校表和招生简章这两个表爬取数据所用的时间在10分钟左右。除了这两个数据量较少的表，其他四个数据量均大于30000的表，每个表的爬取时间都在5个小时以上，而且这个时间还是在数据正常爬取，没有出现任何问题的基础上。如果在爬取过程中出现了超时异常、或者因为一些特殊数据而出现了报错，那么数据爬取则要从头开始，用的时间也会相应增加。

## 4.4系统的发布

研信网网站系统的发布，就是将整个项目部署到服务器上的过程。首先，把数据全部存放在阿里云服务器的数据库里。然后，在JavaWeb后端项目已经写好后，把后端项目部署到阿里云服务器上，并且把各项数据信息通过接口传递给前端，前端项目访问接口，获取数据并展示到网站页面。然后前端项目也部署到阿里云服务器，并且提供一个访问前端页面的URL，项目就部署完成了。已经部署好的项目，只需要在浏览器中输入URL即可随时随地访问。

# 第5章 结束语

20世纪最重要的资源是石油，而21世纪这一状况已经改变了。Google 、脸书、亚马逊掌握着每一个人的信息、行踪、喜好，海量的数据带来巨大的财富和权利。信息时代的竞争已经完全改变，反垄断需要新的标准和紧跟时代的行动。数据经济中谁掌握了数据谁就掌握了市场。

而研信网的开发设计中，考研院校信息的数据是必不可少的。我在用Python爬虫技术爬取数据的过程中，也遇到了很多困难。必须一些容易解决的困难的就是，数据库中字符的长度设计的比较短，因为没有想到一些学校、专业和院校隶属的名字会那么长。还有一些比较难以解决的困难，比如在爬取专业目录的时候，专业目录的数据在table标签里，而table标签里的数据通过普通的etree解析获取不到，只能通过爬取JS里的数据来获取专业目录信息数据，最终在小组另外两成员季波和葛文杰的帮助下，用js2xml解析的方式成功获取了专业目录的数据。在爬取考试范围和分数线的过程中还遇到了一些令人哭笑不得的问题，考试范围和分数线的数据在专业目录的查看详情里，而查看详情的链接无法直接获取，经过一次又一次的寻找与探索，甚至想过怎么直接爬取table标签里的数据，最终偶然间发现，查看详情的链接有一个规律，可以通过js2xml解析获取的数据再经过字符串拼接获得，获得查看详情的链接后，考试范围和分数线的数据就很容易获取了。在数据获取的过程中，还遇到了上述情况以外的许多其他情况，但是在我和小组成员的共同努力下，都一一克服困了难解决了问题。

研信网在我们小组所有成员的共同努力下，取得了阶段性的成果，我个人也较好完成了所负责的数据获取方面的工作。研信网的核心功能是数据，所以研信网的设计开发过程是一个不断完善信息的过程，以上所展示的成果只是我们研信网第一阶段的成果。我们在完成第一阶段的雏形设计后，还会继续做第二阶段的研信网。下一阶段，我们的数据会更详细更充分，界面也更美观大方、符合用户习惯。我们也会推出不同于其他研究生考研网站的功能，找到我们研信网的独特亮点，为帮助考研同学顺利的考入理想研究生院校做出一份努力、贡献一份力量。

在这次实践中，我们小组成员都取得了一些提高。我个人的提高就是：①要有规划的做项目。前期要做好需求分析和任务分工，后期根据团队分工做好自己负责的部分。②有什么好的想法要及时记下来，避免遗忘。③多和小组成员交流。④有计划地做好自己负责的部分，有条不紊的一步步抓进度。

这次的实践也让我认识到，数据分析和获取远远比想象的更复杂，首先要在项目开始前期，设计好各个方面的需求和详细的数据库表。另外在实际操作过程中，在爬取数据的时候，所使用的技术和方法要根据所爬取网站的变化而变化。而且不懂的就要问。在开发的前期阶段，要把数据库设计好，整个项目的逻辑要梳理清楚，多做日志记录。然后在后期实际开发中，根据前期设计按部就班的推进开发进度。这样，不仅开发更高效，也能避免在后期走一些不必要的弯路。

在此非常感谢韩老师的指导、季波同学和葛文杰同学的合作与帮助。

# 指导老师评语

成绩：

指导老师签字：

年 月 日

# 附录

## 1.爬取院校信息的主要代码：

**import** requests  
**from** lxml **import** etree   
  
  
**class** DATA:  
 *# 获取院校信息，不包括院校代码uni\_code，返回list，所有院校信息，包括recruit\_link 招生简章主链接，从研招网* **def** university(self):   
 urls = [**'https://yz.chsi.com.cn/sch/?start={}'**.format(str(i)) **for** i **in** range(0, 841, 20)]  
 **for** page, url **in** enumerate(urls):  
 headers = {**"User-Agent"**:**"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 response = requests.get(url, headers=headers)  
 response\_json\_str = response.content.decode()  
 response\_html = etree.HTML(response\_json\_str) *# type(response\_html) <class 'lxml.etree.\_Element'>* div\_arr = response\_html.xpath(**"//tbody/tr"**)  
 print(**"第"**, page+1, **"页"**)  
 **for** i, div\_one **in** enumerate(div\_arr):  
 uni\_id = page\*20+i+1  
 uni\_name = div\_one.xpath(**"./td[1]/a/text()"**)[0].split()[0]  
 uni\_location = div\_one.xpath(**"./td[2]/text()"**)[0]  
 uni\_subjection = div\_one.xpath(**"./td[3]/text()"**)[0]  
 uni\_characteristics = div\_one.xpath(**"./td[4]/span/text()"**)  
 uni\_characteristic = **''  
 for** f **in** uni\_characteristics:  
 uni\_characteristic = uni\_characteristic+f+**' '** uni\_graduate\_schools = div\_one.xpath(**"./td[5]/i/text()"**) *# graduate\_schools 研究生院* **if** len(uni\_graduate\_schools) == 1:  
 uni\_graduate\_school = **True  
 else**:  
 uni\_graduate\_school = **False** uni\_self\_lines = div\_one.xpath(**"./td[6]/i/text()"**)  
 **if** len(uni\_self\_lines) == 1:  
 uni\_self\_line = **True  
 else**:  
 uni\_self\_line = **False**

## 2.把院校信息数据插入院校表的主要代码：

**import** pymysql  
  
  
**class** DAO:  
 **def** \_\_init\_\_(self):  
 **pass  
  
 def** add(self, uni\_id, uni\_code, uni\_name, uni\_location, uni\_subjection, uni\_characteristic, uni\_graduate\_school, uni\_self\_line):  
 *# 打开数据库连接* db = self.get\_conn()  
 *# 使用cursor()方法获取操作游标* cursor = db.cursor()  
 *# SQL 插入语句* sql = **"INSERT INTO university VALUES ('%d', '%s', '%s', '%s', '%s', '%s', '%d', '%d')"** % \  
 (uni\_id, uni\_code, uni\_name, uni\_location, uni\_subjection, uni\_characteristic, uni\_graduate\_school, uni\_self\_line)  
 **try**:  
 *# 执行sql语句* cursor.execute(sql)  
 *# 执行sql语句* db.commit()  
 print(**"insert ok"**)  
 **except**:  
 *# 发生错误时回滚* print(**"((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((("**)  
 db.rollback()  
 *# 关闭数据库连接* db.close()  
  
 *# 与数据库建立连接* **def** get\_conn(self):  
  
 *# 里皮的阿里云服务器  
 # conn = pymysql.connect(host='39.105.39.215', port=3307, user="root", passwd="qq1040256886", db="yanxin", charset="utf8")  
  
 # 本机的数据库* conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**"root"**, passwd=**"root"**, db=**"yanxin"**, charset=**"utf8"**)  
 **return** conn  
  
 *# 查询并输出所有数据* **def** sel\_all(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from university'** rows = cursor.execute(sql)  
 res = cursor.fetchall()  
 print(**'共有'**, rows, **'条数据'**)  
 **for** re **in** res:  
 print(re)  
 print(**'共有'**, rows, **'条数据'**)**def** is\_empty(self):  
 cursor = self.get\_conn().cursor()  
 sql=**'select \* from university'** rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True****def** del\_all(self):  
 **if** self.is\_empty():  
 print(**"数据为空"**)  
 **else**:  
 sql = **"delete from university"** conn = self.get\_conn()  
 conn.cursor().execute(sql)  
 conn.commit()  
 print(**"已清空数据"**)

## 3.爬取招生简章的主要代码：

**import** requests  
**from** lxml **import** etree  
**from** YanXin.DAO.DAO\_recruit\_page **import** DAO\_recruit\_page  
**import** socket  
  
**class** DATA\_recruit\_page:  
 *# 爬取所有学校的招生简章链接* **def** university\_link(self):  
 urls = [**'http://college.wendu.com/index.php?m=university&c=search&a=search&page={}'**.format(str(i)) **for** i **in** range(1, 70)]  
 headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 **for** url **in** urls:  
 print(url)  
 response = requests.get(url, headers=headers)  
 response\_html = etree.HTML(response.content.decode())  
 schools = response\_html.xpath(**"//a[@class='school\_name']"**)  
 **for** school **in** schools:  
 school\_name = school.xpath(**"./text()"**)[0]  
 school\_link = **"http://college.wendu.com"** + school.xpath(**"./@href"**)[0]  
 self.recruit\_page(school\_link)  
 *# 根据学校链接，爬取一个学校的招生简章信息，并存入数据库* **def** recruit\_page(self, url):  
 headers = {**"User-Agent"**:**"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_html = etree.HTML(response.content.decode())  
 **try**:  
 dao\_recruit\_page = DAO\_recruit\_page()  
 rec\_uni\_name = response\_html.xpath(**"//div[@class='main\_l fl ']/h2/text()"**)[0].split()[0]  
 **if** dao\_recruit\_page.not\_uni\_name(rec\_uni\_name):  
 rec\_year = response\_html.xpath(**"//li[2]/div/h3/div/div[1]/text()"**)[0]  
 contents = response\_html.xpath(**"//li[2]/div/div//p/text()"**)  
 rec\_content = **''  
 for** conts **in** contents:  
 **if** len(conts.split()) > 0:  
 rec\_cont = **''  
 for** cont **in** conts.split():  
 rec\_cont = rec\_cont+cont  
 rec\_content = rec\_content+rec\_cont+**'<br>'** print(rec\_uni\_name)  
 dao\_recruit\_page.add(rec\_year, rec\_content, rec\_uni\_name)  
 **else**:  
 print(rec\_uni\_name, **"已存在"**)  
 **except**:  
 print(**"此处出错"**)

## 4.把招生简章数据插入招生简章表的主要代码：

**import** pymysql  
  
**class** DAO\_recruit\_page:  
 **def** \_\_init\_\_(self):  
 **pass  
  
 def** add(self, rec\_year, rec\_content, rec\_uni\_name):  
 *# 打开数据库连接* db = self.get\_conn()  
 *# 使用cursor()方法获取操作游标* cursor = db.cursor()  
 *# SQL 插入语句* sql = **"INSERT INTO recruit\_page VALUES (null, '%s', '%s', '%s')"** % (rec\_year, rec\_content, rec\_uni\_name)  
 **try**:  
 *# 执行sql语句* cursor.execute(sql)  
 *# 执行sql语句* db.commit()  
 print(**"insert ok"**)  
 **except** Exception **as** e:  
 *# 发生错误时回滚* print(e)  
 print(**"((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((("**)  
 db.rollback()  
 *# 关闭数据库连接* db.close()  
  
 *# 与数据库建立连接* **def** get\_conn(self):  
  
 *# 里皮的阿里云服务器  
 # conn = pymysql.connect(host='39.105.39.215', port=3307, user="root", passwd="qq1040256886", db="yanxin", charset="utf8")  
  
 # 本机的数据库* conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**"root"**, passwd=**"root"**, db=**"yanxin"**, charset=**"utf8"**)  
 **return** conn  
  
 *# 查询并输出所有数据* **def** sel\_all(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from recruit\_page'** rows = cursor.execute(sql)  
 res = cursor.fetchall()  
 print(**'共有'**, rows, **'条数据'**)  
 **for** re **in** res:  
 print(re)  
 print(**'共有'**, rows, **'条数据'**)  
  
 *# 查询表是否为空，为空返回true，不为空返回false，  
 # 用于truncate()* **def** is\_empty(self):  
 cursor = self.get\_conn().cursor()  
 sql=**'select \* from recruit\_page'** rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 查询学校招生简章是否已经爬取，已经爬取返回false，没爬取返回true* **def** not\_uni\_name(self, rec\_uni\_name):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from recruit\_page where rec\_uni\_name = "%s"'** % rec\_uni\_name  
 rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 清空recruit\_page表* **def** truncate(self):  
 **if** self.is\_empty():  
 print(**"数据为空"**)  
 **else**:  
 sql = **"truncate recruit\_page"** conn = self.get\_conn()  
 conn.cursor().execute(sql)  
 conn.commit()  
 print(**"已清空数据"**)

## 5.爬取专业目录的主要代码：

**import** requests  
**from** lxml **import** etree  
**import** js2xml  
**from** YanXin.DAO.DAO\_major\_catalog **import** DAO\_major\_catalog  
  
**class** DATA\_major\_catalog:  
 *# 爬取所有学校的专业链接* **def** major\_catalog\_link(self):  
 urls = [**'http://college.wendu.com/index.php?m=university&c=search&a=search&page={}'**.format(str(i)) **for** i **in** range(1, 70)]  
 headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 **for** url **in** urls:  
 print(url)  
 response = requests.get(url, headers=headers)  
 response\_html = etree.HTML(response.content.decode())  
 schools = response\_html.xpath(**"//a[@class='school\_name']"**)  
 **for** school **in** schools:  
 school\_name = school.xpath(**"./text()"**)[0]  
 major\_link = **"http://college.wendu.com/major\_list-"** + school.xpath(**"./@href"**)[0].split(**"-"**)[1]+**"-1-a-a"** print(school\_name, major\_link)  
 self.major\_catalog(major\_link)  
  
 **def** major\_catalog(self, url):  
 *# url = 'http://college.wendu.com/major\_list-1-1-a-a'* headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 response = requests.get(url, headers=headers)  
 response\_str = response.content.decode()  
 response\_html = etree.HTML(response\_str)  
 script\_list = response\_html.xpath(**"//script/text()"**)  
 *# print(script\_list[6])* **try**:  
 script\_xml = js2xml.parse(script\_list[6], encoding=**'utf-8'**, debug=**False**)  
 *# print(script\_xml)* script\_tree = js2xml.pretty\_print(script\_xml)  
 *# print(script\_tree)* script\_data\_html = etree.HTML(script\_tree)  
 div\_all = script\_data\_html.xpath(**'//var[@name="datas"]/array/object'**)  
 **for** div\_one **in** div\_all:  
 maj\_code = div\_one.xpath(**'.//property[@name="major\_code"]/string/text()'**)[0]  
 maj\_name = div\_one.xpath(**'.//property[@name="major\_name"]/string/text()'**)[0]  
 maj\_college\_name = div\_one.xpath(**'.//property[@name="academy\_name"]/string/text()'**)[0]  
 maj\_uni\_name = response\_html.xpath(**'//p[@class="where"]/a[2]/text()'**)[0]  
 dao = DAO\_major\_catalog()  
 print(maj\_code, maj\_name, maj\_college\_name, maj\_uni\_name)  
 dao.add(maj\_code, maj\_name, maj\_college\_name, maj\_uni\_name)  
 **except**:  
 print(**"有错误》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》"**)

## 6.把专业目录数据插入专业目录表的主要代码：

**import** pymysql  
  
  
**class** DAO\_major\_catalog:  
 **def** \_\_init\_\_(self):  
 **pass** *# 往表 major\_catalog 中添加数据，其中年全* **def** add(self, maj\_code, maj\_name, maj\_college\_name, maj\_uni\_name):  
 *# 打开数据库连接* db = self.get\_conn()  
 *# 使用cursor()方法获取操作游标* cursor = db.cursor()  
 *# SQL 插入语句* sql = **"INSERT INTO major\_catalog VALUES (null, '2018', '%s', '%s', '%s', '%s', '学硕', 3, 0)"** % (maj\_code, maj\_name, maj\_college\_name, maj\_uni\_name)  
 **try**:  
 *# 执行sql语句* cursor.execute(sql)  
 *# 执行sql语句* db.commit()  
 print(**"insert ok"**)  
 **except**:  
 *# 发生错误时回滚* print(**"((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((("**)  
 db.rollback()  
 *# 关闭数据库连接* db.close()  
  
 *# 与数据库建立连接* **def** get\_conn(self):  
  
 *# 里皮的阿里云服务器  
 # conn = pymysql.connect(host='39.105.39.215', port=3307, user="root", passwd="qq1040256886", db="yanxin", charset="utf8")  
  
 # 本机的数据库* conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**"root"**, passwd=**"root"**, db=**"yanxin"**, charset=**"utf8"**)  
 **return** conn  
  
 *# 查询并输出所有数据* **def** sel\_all(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from major\_catalog'** rows = cursor.execute(sql)  
 res = cursor.fetchall()  
 print(**'共有'**, rows, **'条数据'**)  
 **for** re **in** res:  
 print(re)  
 print(**'共有'**, rows, **'条数据'**)  
  
 *# 查询表是否为空，为空返回true，不为空返回false，  
 # 用于truncate()* **def** is\_empty(self):  
 cursor = self.get\_conn().cursor()  
 sql=**'select \* from major\_catalog'** rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 清空 major\_catalog 表* **def** truncate(self):  
 **if** self.is\_empty():  
 print(**"数据为空"**)  
 **else**:  
 sql = **"truncate major\_catalog"** conn = self.get\_conn()  
 conn.cursor().execute(sql)  
 conn.commit()  
 print(**"已清空数据"**)

## 7.爬取专业研究方向的主要代码：

**import** requests  
**import** js2xml  
**from** lxml **import** etree  
**import** socket  
**from** YanXin.DAO.DAO\_research\_direction **import** DAO\_research\_direction  
**class** DATA\_research\_direction:  
 *# 爬取所有学校的专业链接* **def** major\_catalog\_link(self):  
 urls = [**'http://college.wendu.com/index.php?m=university&c=search&a=search&page={}'**.format(str(i)) **for** i **in** range(1, 70)]  
 headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 **for** url **in** urls:  
 print(url)  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_html = etree.HTML(response.content.decode())  
 schools = response\_html.xpath(**"//a[@class='school\_name']"**)  
 **for** school **in** schools:  
 school\_name = school.xpath(**"./text()"**)[0]  
 major\_link = **"http://college.wendu.com/major\_list-"** + school.xpath(**"./@href"**)[0].split(**"-"**)[1]+**"-1-a-a"** print(school\_name, major\_link)  
 self.details\_link(major\_link)  
 *# 爬取某一个学校的专业的查看详情链接* **def** details\_link(self, url):  
 *# url = 'http://college.wendu.com/major\_list-1-1-a-a'* headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_str = response.content.decode()  
 response\_html = etree.HTML(response\_str)  
 script\_list = response\_html.xpath(**"//script/text()"**)  
 *# print(script\_list[6])* **try**:  
 script\_xml = js2xml.parse(script\_list[6], encoding=**'utf-8'**, debug=**False**)  
 *# print(script\_xml)* script\_tree = js2xml.pretty\_print(script\_xml)  
 *# print(script\_tree)* script\_data\_html = etree.HTML(script\_tree)  
 div\_all = script\_data\_html.xpath(**'//var[@name="datas"]/array/object'**)  
 **for** div\_one **in** div\_all:  
 unvid = div\_one.xpath(**'.//property[@name="unvid"]/string/text()'**)[0]  
 acadid = div\_one.xpath(**'.//property[@name="acadid"]/string/text()'**)[0]  
 majorid = div\_one.xpath(**'.//property[@name="majorid"]/string/text()'**)[0]  
 details\_link = **"http://college.wendu.com/major\_detail-"** + unvid + **"-"** + acadid + **"-"** + majorid  
 maj\_name = div\_one.xpath(**'.//property[@name="major\_name"]/string/text()'**)[0]  
 maj\_uni\_name = response\_html.xpath(**'//p[@class="where"]/a[2]/text()'**)[0]  
 print(details\_link, maj\_name, maj\_uni\_name)  
 self.research\_direction(details\_link)  
 **except**:  
 print(**"爬取查看详情链接有错误》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》"**)  
 *# 在查看详情页面爬取 研究方向 具体数据，并插入数据库表research\_direction* **def** research\_direction(self, url):  
 *# url = "http://college.wendu.com/major\_detail-1-122-2"* headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_str = response.content.decode()  
 response\_html = etree.HTML(response\_str)  
 res\_maj\_name = response\_html.xpath(**"//p[@class='where']/text()"**)[2]  
 res\_college\_name = response\_html.xpath(**"//ul[@class='clearfix']/li[2]/span/text()"**)[0]  
 res\_uni\_name = response\_html.xpath(**"//ul[@class='clearfix']/li[1]/span/text()"**)[0]  
 research\_direction\_one = response\_html.xpath(**"//ul[@class='jb\_content']/li[2]/div[@class='menu']/div"**)[0]  
 res\_year = response\_html.xpath(**"//ul[@class='jb\_content']/li[3]/ul/li[1]/text()"**)[0].split(**"："**)[1]  
 res\_register\_num = response\_html.xpath(**"//ul[@class='jb\_content']/li[3]/ul/li[3]/text()"**)[0].split(**"："**)[1]  
 res\_enroll\_num = response\_html.xpath(**"//ul[@class='jb\_content']/li[3]/ul/li[2]/text()"**)[0].split(**"："**)[1]  
 res\_recommend\_num = response\_html.xpath(**"//ul[@class='jb\_content']/li[3]/ul/li[5]/text()"**)[0].split(**"："**)[1]  
 r\_d\_div\_list = research\_direction\_one.xpath(**"./text()"**)  
 r\_d\_span\_list = research\_direction\_one.xpath(**"./span/text()"**)  
 r\_d\_p\_list = research\_direction\_one.xpath(**"./p/text()"**)  
 res\_name = **""** res\_name\_split = **"<br>"** *# print("div", r\_d\_div\_list)  
 # print("span", r\_d\_span\_list)  
 # print("p", r\_d\_p\_list)* r\_d\_div\_one\_one = **""  
 for** r\_d\_div\_one **in** r\_d\_div\_list:  
 **if** len(r\_d\_div\_one.split()) == 0:  
 **continue  
 for** r\_d\_div\_one\_ **in** r\_d\_div\_one.split():  
 r\_d\_div\_one\_one = r\_d\_div\_one\_one+r\_d\_div\_one\_  
 r\_d\_div\_one = r\_d\_div\_one\_one  
 res\_name = res\_name + r\_d\_div\_one + res\_name\_split  
 **if** len(r\_d\_span\_list) > 0:  
 **for** r\_d\_span\_one **in** r\_d\_span\_list:  
 res\_name = res\_name + r\_d\_span\_one + res\_name\_split  
 **if** len(r\_d\_p\_list) > 0:  
 **for** r\_d\_p\_one **in** r\_d\_p\_list:  
 res\_name = res\_name + r\_d\_p\_one + res\_name\_split  
 *# print(res\_name)* **if** res\_name **is not ""**:  
 **try**:  
 res\_way = res\_name.split(**")"**)[0].split(**"("**)[1]  
 **except**:  
 res\_way = **""** dao = DAO\_research\_direction()  
 print(res\_year, res\_name, res\_maj\_name, res\_college\_name, res\_uni\_name, res\_way, res\_register\_num, res\_enroll\_num, res\_recommend\_num)  
 dao.add(res\_year, res\_name, res\_maj\_name, res\_college\_name, res\_uni\_name, res\_way, res\_register\_num, res\_enroll\_num, res\_recommend\_num)

## 8.把专业研究方向数据插入研究方向表的主要代码：

**import** pymysql  
  
**class** DAO\_research\_direction:  
 **def** \_\_init\_\_(self):  
 **pass  
  
 def** add(self, res\_year, res\_name, res\_maj\_name, res\_college\_name, res\_uni\_name, res\_way, res\_register\_num, res\_enroll\_num, res\_recommend\_num):  
 *# 打开数据库连接* db = self.get\_conn()  
 *# 使用cursor()方法获取操作游标* cursor = db.cursor()  
 *# SQL 插入语句* sql = **"INSERT INTO research\_direction VALUES (null, '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s')"** % (res\_year, res\_name, res\_maj\_name, res\_college\_name, res\_uni\_name, res\_way, res\_register\_num, res\_enroll\_num, res\_recommend\_num)  
 **try**:  
 *# 执行sql语句* cursor.execute(sql)  
 *# 执行sql语句* db.commit()  
 print(**"insert ok"**)  
 **except** Exception **as** e:  
 *# 发生错误时回滚* print(e)  
 print(**"((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((("**)  
 db.rollback()  
 *# 关闭数据库连接* db.close()  
  
 *# 与数据库建立连接* **def** get\_conn(self):  
  
 *# 里皮的阿里云服务器  
 # conn = pymysql.connect(host='39.105.39.215', port=3307, user="root", passwd="qq1040256886", db="yanxin", charset="utf8")  
  
 # 本机的数据库* conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**"root"**, passwd=**"root"**, db=**"yanxin"**, charset=**"utf8"**)  
 **return** conn  
  
 *# 查询并输出所有数据* **def** sel\_all(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from research\_direction'** rows = cursor.execute(sql)  
 res = cursor.fetchall()  
 print(**'共有'**, rows, **'条数据'**)  
 **for** re **in** res:  
 print(re)  
 print(**'共有'**, rows, **'条数据'**)  
  
 *# 查询表是否为空，为空返回true，不为空返回false，  
 # 用于truncate()* **def** is\_empty(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from research\_direction'** rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 清空recruit\_page表* **def** truncate(self):  
 **if** self.is\_empty():  
 print(**"数据为空"**)  
 **else**:  
 sql = **"truncate research\_direction"** conn = self.get\_conn()  
 conn.cursor().execute(sql)  
 conn.commit()  
 print(**"已清空数据"**)

## 9.爬取专业考试范围的主要代码：

**import** requests  
**import** js2xml  
**import** socket  
**from** lxml **import** etree  
**from** YanXin.DAO.DAO\_exam\_area **import** DAO\_exam\_area  
**class** DATA\_exam\_area:  
 *# 爬取所有学校的专业链接* **def** major\_catalog\_link(self):  
 urls = [**'http://college.wendu.com/index.php?m=university&c=search&a=search&page={}'**.format(str(i)) **for** i **in** range(1, 70)]  
 headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 **for** url **in** urls:  
 print(url)  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_html = etree.HTML(response.content.decode())  
 schools = response\_html.xpath(**"//a[@class='school\_name']"**)  
 **for** school **in** schools:  
 school\_name = school.xpath(**"./text()"**)[0]  
 major\_link = **"http://college.wendu.com/major\_list-"** + school.xpath(**"./@href"**)[0].split(**"-"**)[1]+**"-1-a-a"** print(school\_name, major\_link)  
 self.details\_link(major\_link)  
 *# 爬取某一个学校的专业的查看详情链接* **def** details\_link(self, url):  
 *# url = 'http://college.wendu.com/major\_list-1-1-a-a'* headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_str = response.content.decode()  
 response\_html = etree.HTML(response\_str)  
 script\_list = response\_html.xpath(**"//script/text()"**)  
 *# print(script\_list[6])* **try**:  
 script\_xml = js2xml.parse(script\_list[6], encoding=**'utf-8'**, debug=**False**)  
 *# print(script\_xml)* script\_tree = js2xml.pretty\_print(script\_xml)  
 *# print(script\_tree)* script\_data\_html = etree.HTML(script\_tree)  
 div\_all = script\_data\_html.xpath(**'//var[@name="datas"]/array/object'**)  
 **for** div\_one **in** div\_all:  
 unvid = div\_one.xpath(**'.//property[@name="unvid"]/string/text()'**)[0]  
 acadid = div\_one.xpath(**'.//property[@name="acadid"]/string/text()'**)[0]  
 majorid = div\_one.xpath(**'.//property[@name="majorid"]/string/text()'**)[0]  
 details\_link = **"http://college.wendu.com/major\_detail-"** + unvid + **"-"** + acadid + **"-"** + majorid  
 maj\_name = div\_one.xpath(**'.//property[@name="major\_name"]/string/text()'**)[0]  
 maj\_uni\_name = response\_html.xpath(**'//p[@class="where"]/a[2]/text()'**)[0]  
 print(details\_link, maj\_name, maj\_uni\_name)  
 self.exam\_area(details\_link)  
 **except** Exception **as** e:  
 print(**"爬取查看详情链接有错误》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》"**)  
 print(e)  
 *# 在查看详情页面爬取 考试范围 具体数据，并插入数据库表 exam\_area* **def** exam\_area(self, url):  
 *# url = "http://college.wendu.com/major\_detail-143-3504-137"* headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_str = response.content.decode()  
 response\_html = etree.HTML(response\_str)  
 exa\_year = response\_html.xpath(**"//ul[@class='jb\_content']/li[3]/ul/li[1]/text()"**)[0].split(**"："**)[1]  
 exa\_maj\_name = response\_html.xpath(**"//p[@class='where']/text()"**)[2]  
 exa\_college\_name = response\_html.xpath(**"//ul[@class='clearfix']/li[2]/span/text()"**)[0]  
 exa\_uni\_name = response\_html.xpath(**"//ul[@class='clearfix']/li[1]/span/text()"**)[0]  
 exam\_area\_one = response\_html.xpath(**"//ul[@class='jb\_content']/li[5]/div[2]/div"**)[0]  
 e\_a\_div\_list = exam\_area\_one.xpath(**"./text()"**)  
 e\_a\_span\_list = exam\_area\_one.xpath(**"./span/text()"**)  
 e\_a\_p\_list = exam\_area\_one.xpath(**"./p/text()"**)  
 *# print("div", e\_a\_div\_list)  
 # print("span", e\_a\_span\_list)  
 # print("p", e\_a\_p\_list)* exa\_subjects = **""** exa\_subject\_list = []  
 e\_a\_div\_one\_one = **""** dao = DAO\_exam\_area()  
 *# e\_a\_div\_one ： div集合的一个元素  
 # e\_a\_div\_one\_ ： div集合的一个元素,用空格符换行符分割后的集合元素（目的为去除换行符）  
 # e\_a\_div\_one\_one ： div集合的一个元素（去除换行符）  
 # 下面的for if if内容的作用：把div、span、p的集合元素全部集中到 exa\_subject\_list 中* **for** e\_a\_div\_one **in** e\_a\_div\_list:  
 *# 如果div标签的 一个元素 里只有换行符，则不拼接它* **if** len(e\_a\_div\_one.split()) == 0:  
 **continue  
 for** e\_a\_div\_one\_ **in** e\_a\_div\_one.split():  
 e\_a\_div\_one\_one = e\_a\_div\_one\_one + e\_a\_div\_one\_  
 exa\_subject\_list.append(e\_a\_div\_one\_one)  
 **if** len(e\_a\_span\_list) > 0:  
 **for** e\_a\_span\_one **in** e\_a\_span\_list:  
 exa\_subject\_list.append(e\_a\_span\_one)  
 **if** len(e\_a\_p\_list) > 0:  
 **for** e\_a\_p\_one **in** e\_a\_p\_list:  
 exa\_subject\_list.append(e\_a\_p\_one)  
 *# if内容的作用：把集合 exa\_subject\_list 字符串拼接到 exa\_subjects 中* **if** len(exa\_subject\_list) > 0:  
 **for** exa\_subject\_one **in** exa\_subject\_list:  
 exa\_subjects = exa\_subjects+exa\_subject\_one  
 print(**"exa\_subject\_list"**, exa\_subject\_list)  
 *# print("exa\_subjects", exa\_subjects)  
 # if内容的作用：分离出四科* **if** len(exa\_subjects.split(**"①"**)) == 2 **and** len(exa\_subject\_list) >= 1 **and** len(exa\_subject\_list) <= 5 :  
 exa\_subjects = **""  
 for** i, e\_s\_l\_one **in** enumerate(exa\_subject\_list):  
 **if** i < 4:  
 exa\_subjects = exa\_subjects + e\_s\_l\_one  
 **if** len(exa\_subjects.split(**"④"**)) == 2:  
 exa\_subject1 = exa\_subjects.split(**"①"**)[1].split(**"②"**)[0]  
 exa\_subject2 = exa\_subjects.split(**"②"**)[1].split(**"③"**)[0]  
 exa\_subject3 = exa\_subjects.split(**"③"**)[1].split(**"④"**)[0]  
 exa\_subject4 = exa\_subjects.split(**"④"**)[1]  
 **elif** len(exa\_subjects.split(**"④"**)) == 1:  
 exa\_subject1 = exa\_subjects.split(**"①"**)[1].split(**"②"**)[0]  
 exa\_subject2 = exa\_subjects.split(**"②"**)[1].split(**"③"**)[0]  
 exa\_subject3 = exa\_subjects.split(**"③"**)[1]  
 exa\_subject4 = **""** print(**"exa\_subject1-----"**, exa\_subject1, **"-------"**)  
 print(**"exa\_subject2-----"**, exa\_subject2, **"-------"**)  
 print(**"exa\_subject3-----"**, exa\_subject3, **"-------"**)  
 print(**"exa\_subject4-----"**, exa\_subject4, **"-------"**)  
 print(exa\_year, exa\_maj\_name, exa\_college\_name, exa\_uni\_name, exa\_subject1, exa\_subject2, exa\_subject3, exa\_subject4)  
 dao.add(exa\_year, exa\_maj\_name, exa\_college\_name, exa\_uni\_name, exa\_subject1, exa\_subject2, exa\_subject3, exa\_subject4)  
 *# print("exa\_subjects ：", exa\_subjects)  
 # elif内容的作用：分离不出来的，放到科一* **elif** exa\_subjects **is not "" and** len(exa\_subject\_list) > 0:  
 print(**"1111111111111111111111115555555555555555555555555"**)  
 print(**"exa\_subjects ："**, exa\_subjects)  
 print(exa\_year, exa\_maj\_name, exa\_college\_name, exa\_uni\_name, exa\_subjects)  
 dao.add(exa\_year, exa\_maj\_name, exa\_college\_name, exa\_uni\_name, exa\_subjects, **''**, **''**, **''**)

## 10.把专业考试范围数据插入考试范围表的主要代码：

**import** pymysql  
  
**class** DAO\_exam\_area:  
 **def** \_\_init\_\_(self):  
 **pass  
  
 def** add(self, exa\_year, exa\_maj\_name, exa\_college\_name, exa\_uni\_name, exa\_subject1, exa\_subject2, exa\_subject3, exa\_subject4):  
 *# 打开数据库连接* db = self.get\_conn()  
 *# 使用cursor()方法获取操作游标* cursor = db.cursor()  
 *# SQL 插入语句* sql = **"INSERT INTO exam\_area VALUES (null, '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s')"** % (exa\_year, exa\_maj\_name, exa\_college\_name, exa\_uni\_name, exa\_subject1, exa\_subject2, exa\_subject3, exa\_subject4)  
 **try**:  
 *# 执行sql语句* cursor.execute(sql)  
 *# 执行sql语句* db.commit()  
 print(**"insert ok"**)  
 **except** Exception **as** e:  
 *# 发生错误时回滚* print(e)  
 print(**"((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((("**)  
 db.rollback()  
 *# 关闭数据库连接* db.close()  
  
 *# 与数据库建立连接* **def** get\_conn(self):  
  
 *# 里皮的阿里云服务器  
 # conn = pymysql.connect(host='39.105.39.215', port=3307, user="root", passwd="qq1040256886", db="yanxin", charset="utf8")  
  
 # 本机的数据库* conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**"root"**, passwd=**"root"**, db=**"yanxin"**, charset=**"utf8"**)  
 **return** conn  
  
 *# 查询并输出所有数据* **def** sel\_all(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from exam\_area'** rows = cursor.execute(sql)  
 res = cursor.fetchall()  
 print(**'共有'**, rows, **'条数据'**)  
 **for** re **in** res:  
 print(re)  
 print(**'共有'**, rows, **'条数据'**)  
  
 *# 查询表是否为空，为空返回true，不为空返回false，  
 # 用于truncate()* **def** is\_empty(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from exam\_area'** rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 查询学校 exam\_area 是否已经爬取，已经爬取返回false，没爬取返回true* **def** not\_uni\_name(self, exa\_uni\_name):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from exam\_area where exa\_uni\_name = "%s"'** % exa\_uni\_name  
 rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 清空recruit\_page表* **def** truncate(self):  
 **if** self.is\_empty():  
 print(**"数据为空"**)  
 **else**:  
 sql = **"truncate exam\_area"** conn = self.get\_conn()  
 conn.cursor().execute(sql)  
 conn.commit()  
 print(**"已清空数据"**)

## 11.爬取专业分数线的主要代码：

**import** requests  
**import** js2xml  
**import** socket  
**from** lxml **import** etree  
**from** YanXin.DAO.DAO\_score\_level **import** DAO\_score\_level  
**class** DATA\_score\_level:  
 *# 爬取所有学校的专业链接* **def** major\_catalog\_link(self):  
 urls = [**'http://college.wendu.com/index.php?m=university&c=search&a=search&page={}'**.format(str(i)) **for** i **in** range(1, 70)]  
 headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 **for** url **in** urls:  
 print(url)  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_html = etree.HTML(response.content.decode())  
 schools = response\_html.xpath(**"//a[@class='school\_name']"**)  
 **for** school **in** schools:  
 school\_name = school.xpath(**"./text()"**)[0]  
 major\_link = **"http://college.wendu.com/major\_list-"** + school.xpath(**"./@href"**)[0].split(**"-"**)[1]+**"-1-a-a"** print(school\_name, major\_link)  
 dao = DAO\_score\_level()  
 **if** dao.not\_uni\_name(school\_name):  
 self.details\_link(major\_link)  
 *# 爬取某一个学校的专业的查看详情链接* **def** details\_link(self, url):  
 *# url = 'http://college.wendu.com/major\_list-1-1-a-a'* headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_str = response.content.decode()  
 response\_html = etree.HTML(response\_str)  
 script\_list = response\_html.xpath(**"//script/text()"**)  
 *# print(script\_list[6])* **try**:  
 script\_xml = js2xml.parse(script\_list[6], encoding=**'utf-8'**, debug=**False**)  
 *# print(script\_xml)* script\_tree = js2xml.pretty\_print(script\_xml)  
 *# print(script\_tree)* script\_data\_html = etree.HTML(script\_tree)  
 div\_all = script\_data\_html.xpath(**'//var[@name="datas"]/array/object'**)  
 **for** div\_one **in** div\_all:  
 unvid = div\_one.xpath(**'.//property[@name="unvid"]/string/text()'**)[0]  
 acadid = div\_one.xpath(**'.//property[@name="acadid"]/string/text()'**)[0]  
 majorid = div\_one.xpath(**'.//property[@name="majorid"]/string/text()'**)[0]  
 details\_link = **"http://college.wendu.com/major\_detail-"** + unvid + **"-"** + acadid + **"-"** + majorid  
 maj\_name = div\_one.xpath(**'.//property[@name="major\_name"]/string/text()'**)[0]  
 maj\_uni\_name = response\_html.xpath(**'//p[@class="where"]/a[2]/text()'**)[0]  
 print(details\_link, maj\_name, maj\_uni\_name)  
 self.score\_level(details\_link)  
 **except** Exception **as** e:  
 print(**"爬取查看详情链接有错误》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》"**)  
 print(e)  
 *# 在查看详情页面爬取 考试范围 具体数据，并插入数据库表 exam\_area* **def** score\_level(self, url):  
 *# url = "http://college.wendu.com/major\_detail-143-3504-137"* headers = {**"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36"**}  
 socket.setdefaulttimeout(600)  
 response = requests.get(url, headers=headers, timeout=600)  
 response\_str = response.content.decode()  
 response\_html = etree.HTML(response\_str)  
 sco\_year = response\_html.xpath(**"//ul[@class='jb\_content']/li[4]/ul/li[1]/text()"**)[0].split(**"："**)[1]  
 sco\_maj\_name = response\_html.xpath(**"//p[@class='where']/text()"**)[2]  
 sco\_college\_name = response\_html.xpath(**"//ul[@class='clearfix']/li[2]/span/text()"**)[0]  
 sco\_uni\_name = response\_html.xpath(**"//ul[@class='clearfix']/li[1]/span/text()"**)[0]  
 sco\_reexamine\_line = response\_html.xpath(**"//ul[@class='jb\_content']/li[4]/ul/li[2]/text()"**)[0].split(**"："**)[1]  
 sco\_subject1 = response\_html.xpath(**"//ul[@class='jb\_content']/li[4]/ul/li[3]/text()"**)[0].split(**"："**)[1]  
 sco\_subject2 = response\_html.xpath(**"//ul[@class='jb\_content']/li[4]/ul/li[4]/text()"**)[0].split(**"："**)[1]  
 sco\_subject3 = response\_html.xpath(**"//ul[@class='jb\_content']/li[4]/ul/li[5]/text()"**)[0].split(**"："**)[1]  
 sco\_subject4 = response\_html.xpath(**"//ul[@class='jb\_content']/li[4]/ul/li[6]/text()"**)[0].split(**"："**)[1]  
 *# sco\_subjects作用：检查爬取分数线是否为空* sco\_subjects = sco\_subject1+sco\_subject2+sco\_subject3+sco\_subject4  
 dao = DAO\_score\_level()  
 **if** sco\_subjects **is not ""**:  
 print(sco\_year, sco\_maj\_name, sco\_college\_name, sco\_uni\_name, sco\_reexamine\_line, sco\_subject1, sco\_subject2, sco\_subject3, sco\_subject4)  
 **if** dao.not\_uni\_name\_maj\_name(sco\_uni\_name, sco\_college\_name, sco\_maj\_name):  
 dao.add(sco\_year, sco\_maj\_name, sco\_college\_name, sco\_uni\_name, sco\_reexamine\_line, sco\_subject1, sco\_subject2, sco\_subject3, sco\_subject4)  
 **else**:  
 print(**"数据重复"**)

## 12.把专业分数线数据插入分数线表的主要代码：

**import** pymysql  
  
**class** DAO\_score\_level:  
 **def** \_\_init\_\_(self):  
 **pass  
  
 def** add(self, sco\_year, sco\_maj\_name, sco\_college\_name, sco\_uni\_name,sco\_reexamine\_line, sco\_subject1, sco\_subject2, sco\_subject3, sco\_subject4):  
 *# 打开数据库连接* db = self.get\_conn()  
 *# 使用cursor()方法获取操作游标* cursor = db.cursor()  
 *# SQL 插入语句* sql = **"INSERT INTO score\_level VALUES (null, '%s', '%s', '%s', '%s', 0, 0, '%s', '%s', '%s', '%s', '%s')"** % (sco\_year, sco\_maj\_name, sco\_college\_name, sco\_uni\_name, sco\_reexamine\_line, sco\_subject1, sco\_subject2, sco\_subject3, sco\_subject4)  
 **try**:  
 *# 执行sql语句* cursor.execute(sql)  
 *# 执行sql语句* db.commit()  
 print(**"insert ok"**)  
 **except** Exception **as** e:  
 *# 发生错误时回滚* print(e)  
 print(**"((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((("**)  
 db.rollback()  
 *# 关闭数据库连接* db.close()  
  
 *# 与数据库建立连接* **def** get\_conn(self):  
  
 *# 里皮的阿里云服务器  
 # conn = pymysql.connect(host='39.105.39.215', port=3307, user="root", passwd="qq1040256886", db="yanxin", charset="utf8")  
  
 # 本机的数据库* conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**"root"**, passwd=**"root"**, db=**"yanxin"**, charset=**"utf8"**)  
 **return** conn  
  
 *# 查询并输出所有数据* **def** sel\_all(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from score\_level'** rows = cursor.execute(sql)  
 res = cursor.fetchall()  
 print(**'共有'**, rows, **'条数据'**)  
 **for** re **in** res:  
 print(re)  
 print(**'共有'**, rows, **'条数据'**)  
  
 *# 查询表是否为空，为空返回true，不为空返回false，  
 # 用于truncate()* **def** is\_empty(self):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from score\_level'** rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 查询学校 学院 专业 score\_level 是否已经爬取，已经爬取返回false，没爬取返回true* **def** not\_uni\_name\_maj\_name(self, sco\_uni\_name, sco\_college\_name, sco\_maj\_name):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from score\_level where sco\_uni\_name = "%s" and sco\_college\_name = "%s" and sco\_maj\_name = "%s"'** % (sco\_uni\_name, sco\_college\_name, sco\_maj\_name)  
 rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 查询学校 学院 专业 score\_level 是否已经爬取，已经爬取返回false，没爬取返回true* **def** not\_uni\_name(self, sco\_uni\_name):  
 cursor = self.get\_conn().cursor()  
 sql = **'select \* from score\_level where sco\_uni\_name = "%s"'** % sco\_uni\_name  
 rows = cursor.execute(sql)  
 **if** rows > 0:  
 **return False  
 else**:  
 **return True** *# 清空recruit\_page表* **def** truncate(self):  
 **if** self.is\_empty():  
 print(**"数据为空"**)  
 **else**:  
 sql = **"truncate score\_level"** conn = self.get\_conn()  
 conn.cursor().execute(sql)  
 conn.commit()  
 print(**"已清空数据"**)