**必做实验①：**

使用抓包器，去抓取一次web访问过程中的数据包，分析一下Requests and responses 数据的数据包格式。可以看到在tcp数据包的payload部分是http协议数据包，又分为了headers和body等部分。

发送的Request，最核心的部分是一个url。就是大家通常说的网址。其中/前的部分，是主机地址，会被解析为ip地址。第一个/后的部分为路径和参数。会对应服务器上的文件或者功能。然后，服务器处理完毕了，返回response，而response一般是一个html格式的文本。

结果：

浏览器会直接把html渲染为我们所看到的页面的不同样式。html有自己专门的语法。html的格式是以标签树的结构组织的。

**必做实验②：**

用chrome的开发者工具的 elements、sources和network几个工具，分析网页。

开发一个web程序的基本过程

上周演示的程序使用python内置的库开发了一个基本的http服务器端。

以django为例，演示一下，web开发的主要流程，然后我会说sql注入的基本原理。然后再给大家分析一下sql注入和xss在django框架下是如何被解决的。

参考文档：<https://docs.djangoproject.com/en/3.0/>

**第一步，安装 Django**

pip install Django 或者 python -m pip install Django

这里补充一个知识点，如果pip安装速度很慢，可以修改pip使用的镜像源，改为国内的源，速度就会很快，比如我图里使用的aliyun的镜像。

**第二步，执行**

django-admin startproject mysite

cd mysite

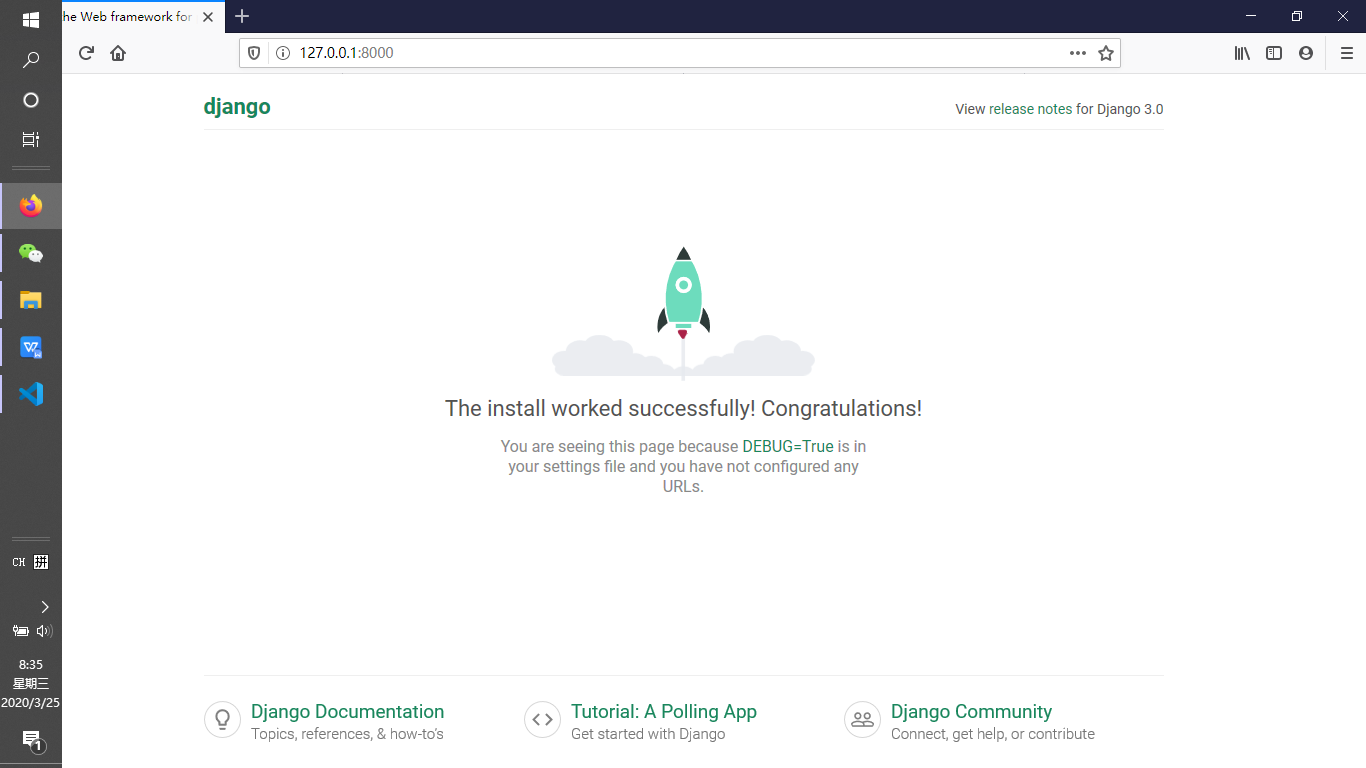
python manage.py startapp polls

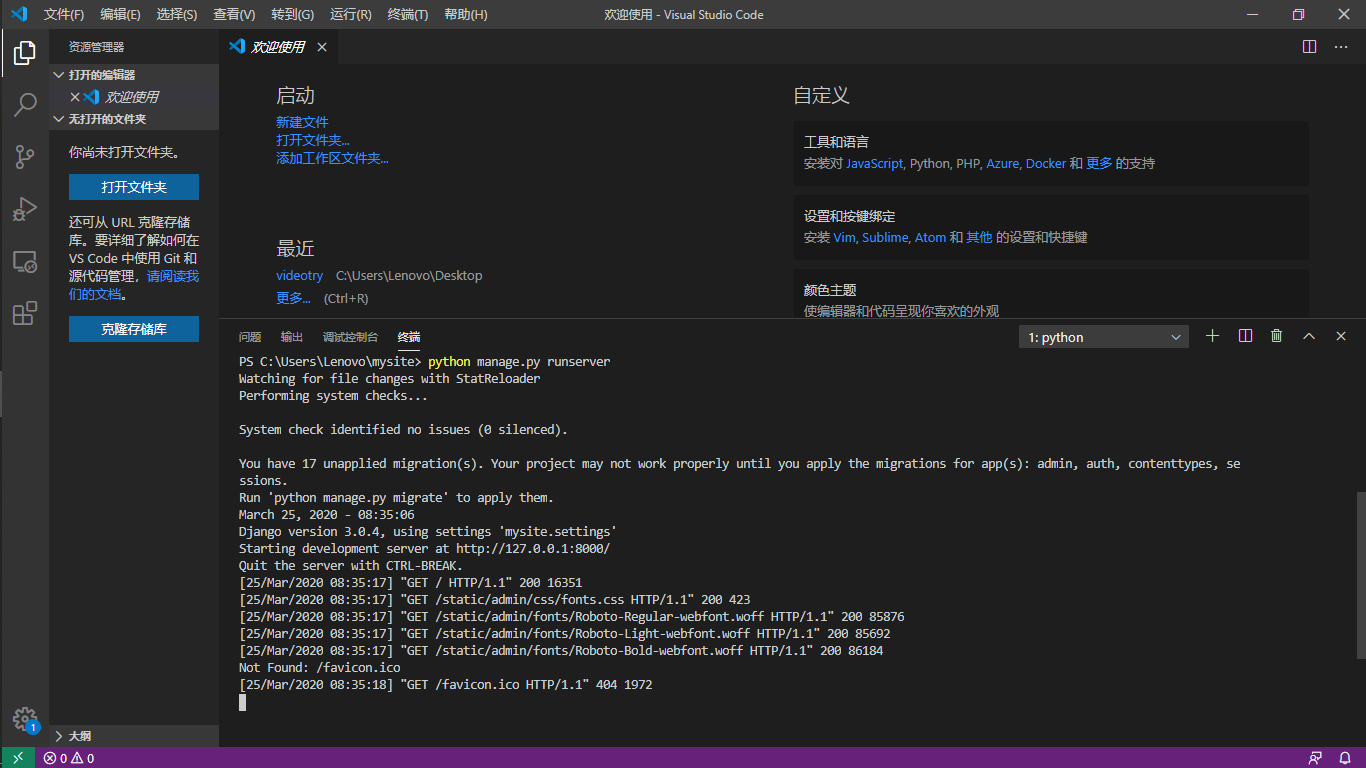
python manage.py runserver

四个命令，构建了一个基于Django的基本框架的web应用程序。然后可以访问 <http://127.0.0.1:8000/> 可以看到结果。

参考文档：<https://docs.djangoproject.com/en/3.0/intro/tutorial01/>

结果：





命令行里，可以看到服务器的打印输出，表示服务器收到了 request。

看到的页面就是框架自动返回给大家的response。说明，request和response，请求相应的处理过程已经搭建起来了。

下面，来做一个简单的教务管理系统。

教务管理系统肯定要用到数据库。

以下简述Django框架的基本编程的结构：

mvc的概念：编写大型程序的时候，一定要做到mvc分离，m数据模型，我们要先建立基础的底层的数据结构。然后在处理用户输入，构建每个用户输入对应的处理函数。就是c 控制。然后，在底层m数据模型的基础上，绘制用户界面。比如写贪吃蛇游戏，最先做的事情，使用链表来保存蛇和食物的相应的数据，写一些处理这个数据的函数，供上层的c和v来调用。我们把这个叫做封装。

这是基本的编程思想，和正确的工作组织流程。大到一个复杂的大型web程序，其实底层思想还是mvc。只是换了个名字，叫mvt。t是页面模板。

写Django的程序，或者任何框架程序。主要就是写三大部分，第一，数据模型，models，第二，views和url。是用户请求对应的处理程序。第三，前端页面模板。处理完的结果如何显示的问题。其中url部分，又称为路由。是将用户请求的url，对应到处理函数的过程。Django的处理函数，有一个专门名称，叫views。其基本过程就是框架收到用户的request ，其中有URL。框架根据urls.py中的配置。将用户请求对应到一个处理函数。一般在views.py中。views.py中的函数，参数就是request对象，Django中是HttpRequest类。然后views函数需要返回一个HTTP Response类型的request，Django把这个reqeust变化为http协议的request数据包，返回给浏览器。一般在views的处理过程中，会访问数据库，也就是models。models把底层的数据库操作，比如sql全部封装为了对象化的处理。比如底层操作数据库是用sql语句，这个大家在数据的课程中学习过。所以我们最原始的web程序，一般会程序员拼装sql语句。但是在Django中不用。我们把这种底层数据的封装，称为orm。Object-relational Mapper。

**数据库搭建：**

Django最方便的一点，就是把建表这种基本操作。编程了python中的类的定义。所定义的这些类，直接继承models.Model。程序员只需要写好这个models.py文件。所有的建表的操作框架就可以完成。

大家可以再建一个app ，比如叫 edu\_admin

执行以下两条指令：

python manage.py startapp edu\_admin

code . //即用vscode打开工程

然后对student进行修改。一共三个表，学生、课程、成绩。其中成绩表就是学生和课程的关系，也就是选课。成绩表中的一个记录，表示某个学生，选了某门课程。

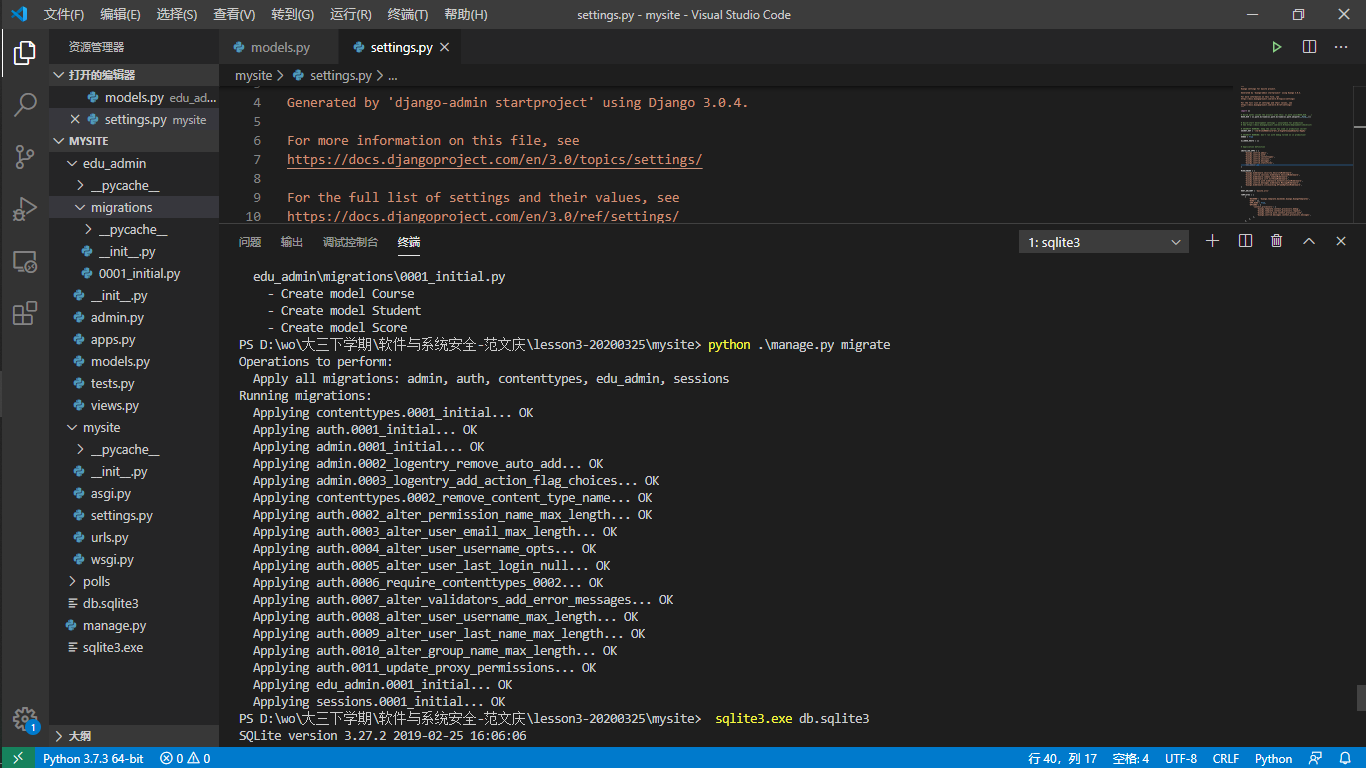
然后，我们需要把这个表结构，真实的写入到数据库中。也就是create table的过程。django称为migrate。大家打开 mysite的settings.py。在 INSTALLED\_APPS 这里增加一个 edu\_admin。表示 edu\_admin 这个是我们这个site的一个app。之前startapp命令只是创建了app，必须要把app写入到这里，这个app才会被纳入到站点功能中。

然后

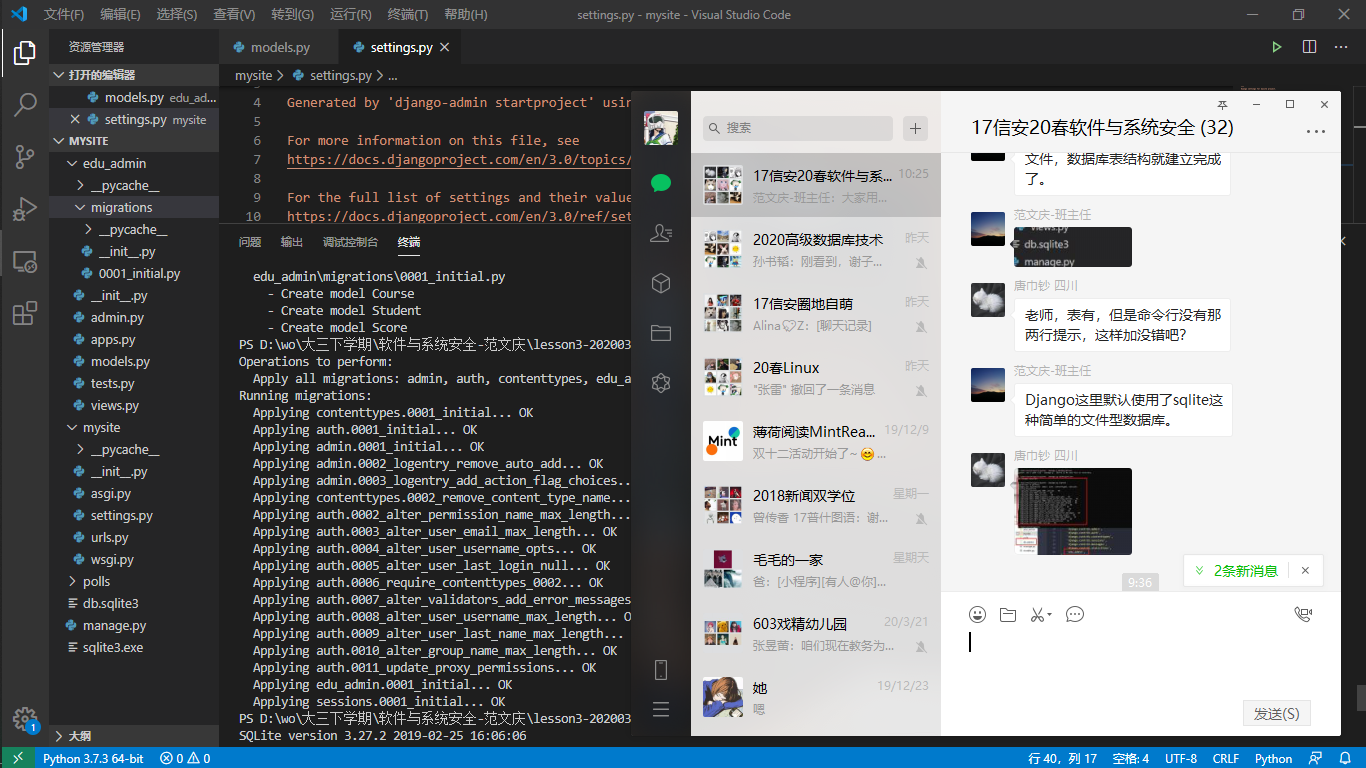
python .\manage.py makemigrations

python .\manage.py migrate

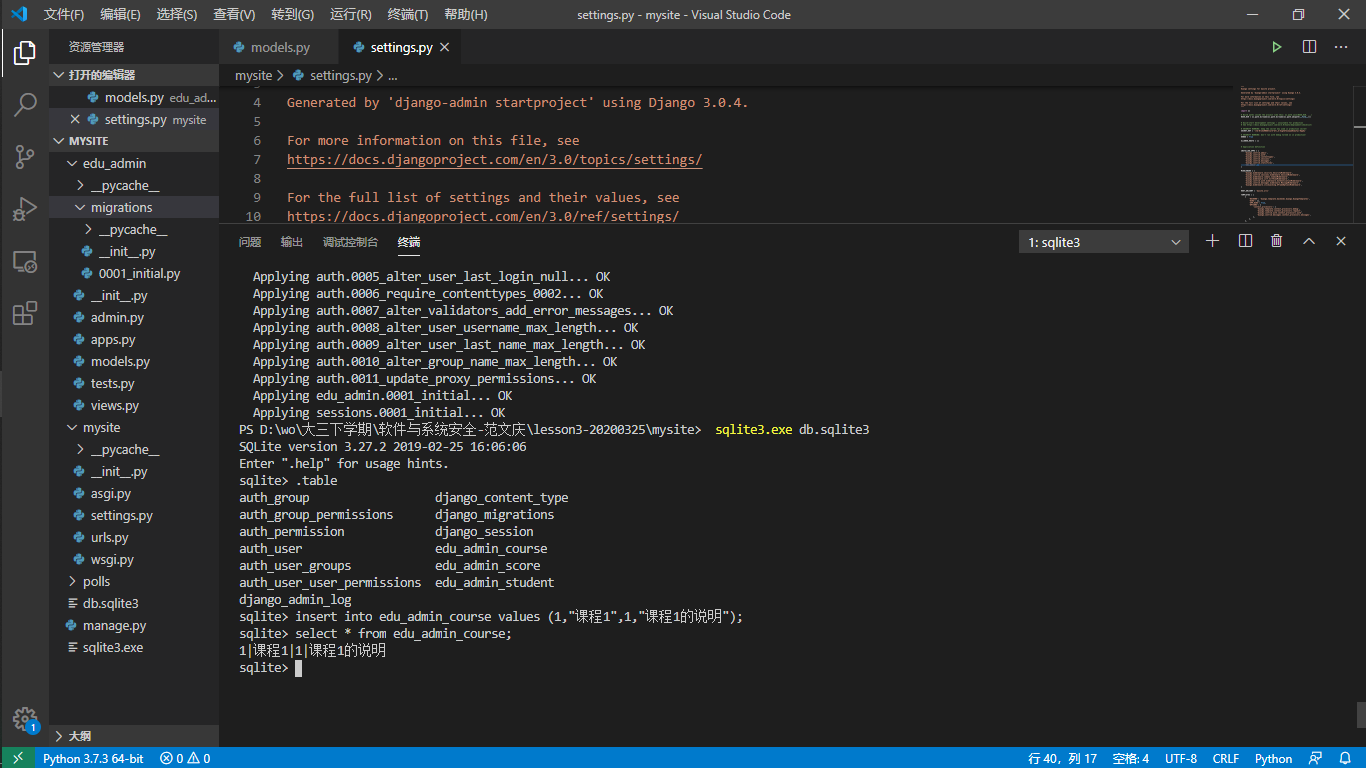
会有这样的结果：



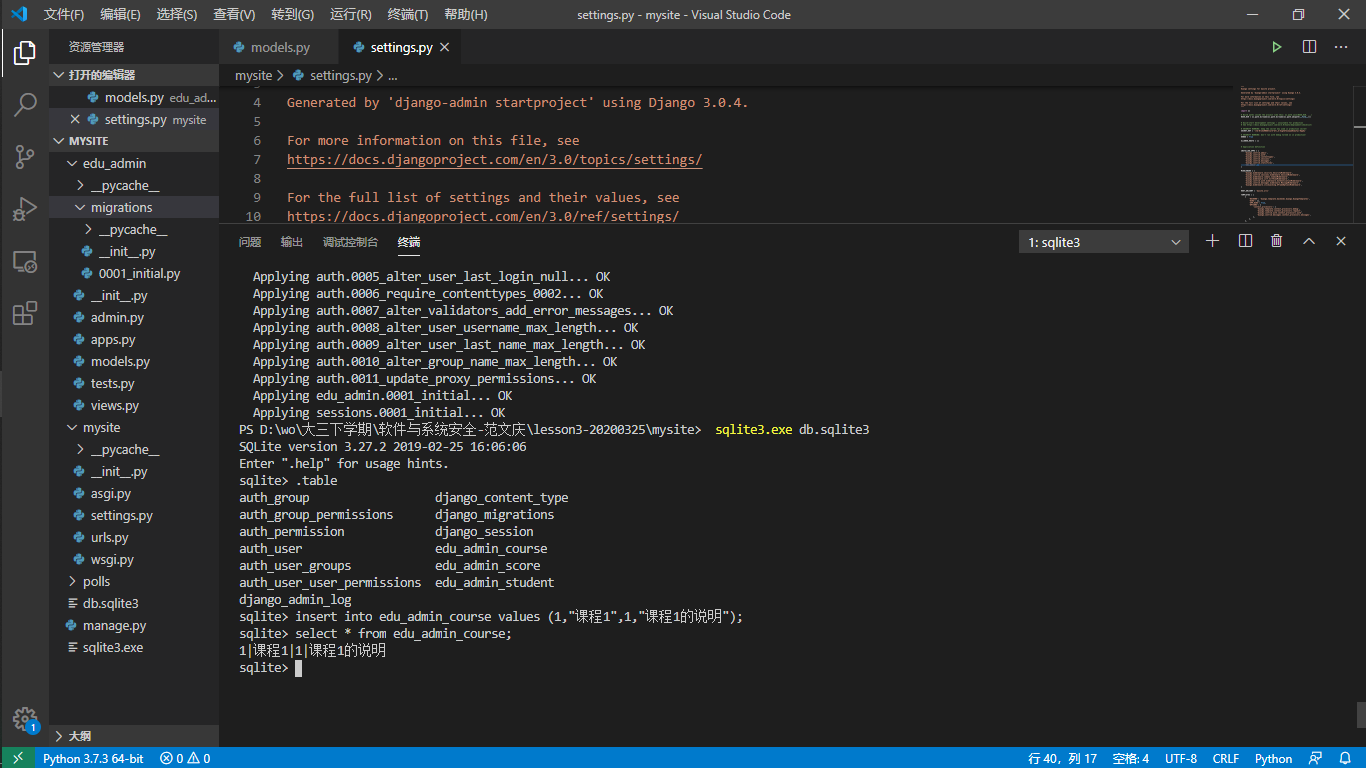
然后会出现一个 db.sqlite3，文件，数据库表结构就建立完成了。



Django这里默认使用了sqlite这种简单的文件型数据库。settings里加app，加了才会有刚才建的表。Django这里默认使用了sqlite这种简单的文件型数据库。这种数据库的好处是不用按照，就是一个文件来保存数据的所有信息，适合轻量级小规模的应用。但是效率和容量都有效。一般用在开发调试环境，不用在生产环境。加了app以后，执行makemigrations和migrate。



makemigrations成功的标志是在app的目录下有migrations目录。



也就是我们这里的edu\_admin

参考资料：<https://www.sqlite.org/download.html>

为了验证Django真的建立了表，我们去下载一个sqlite的客户端软件，来看一下它的表结构。Windows的同学，下载sqlite-tools-win32-x86-3310100.zip。Linux的同学直接 apt install sqlite3。把这个exe加入在PATH环境变量，或者放在db.sqlite同一个目录。然后

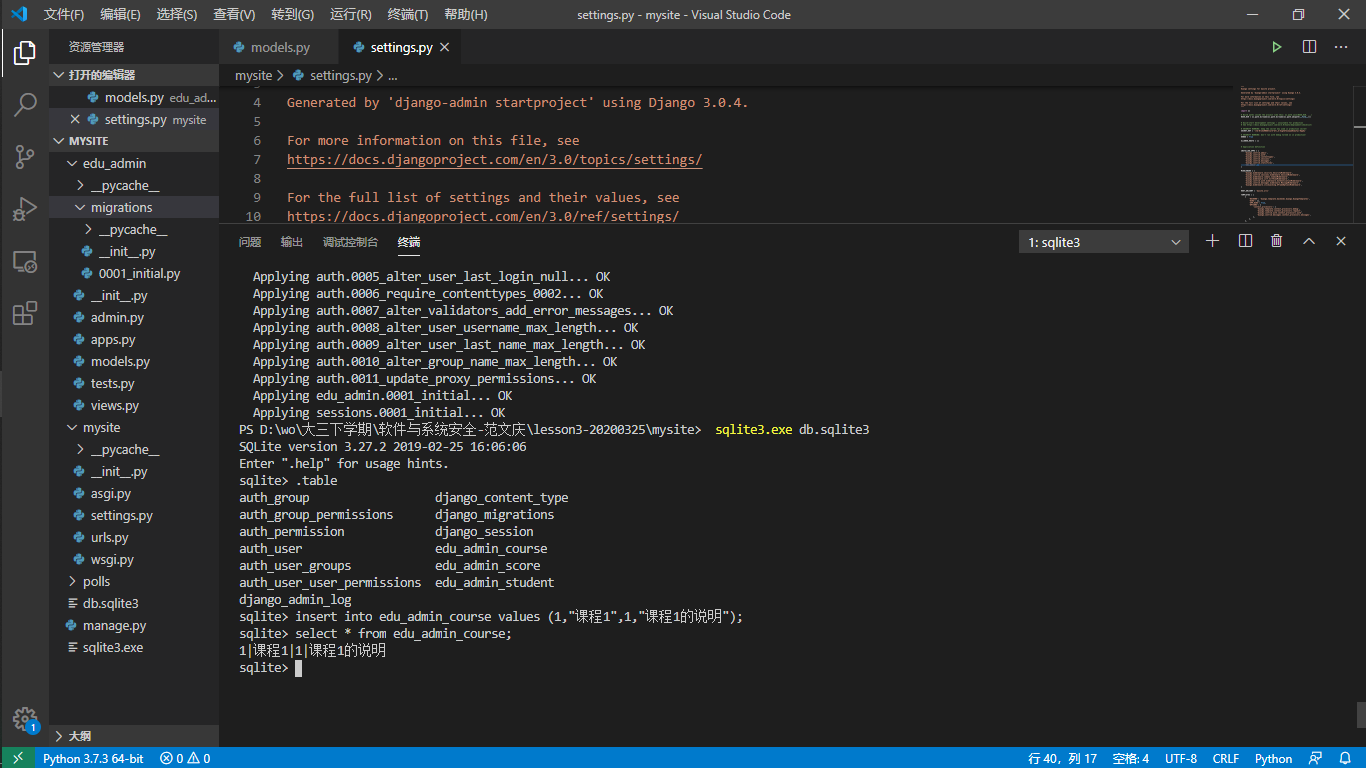
执行命令

sqlite3.exe db.sqlite

进入到sqlite的命令行以后，执行命令：

.table

然后可以看到所有的表，其中以下三个表是我们在models中定义的。其他表是Django自己要用的。



然后大家可以执行sql语句，插入一条记录。insert和select可以成功。说明表是好的。这一步以后，其实已经可以sql注入了。

作业：

在不使用Django的情况下，我们可以使用更底层的python的sqlite库来编程操作数据库。参考文献：[https://docs.python.org/3/library/sqlite3.html](https://docs.python.org/3/library/sqlite3.html。)

大家可以在上周的httpserver.py的基础上，继续编写漏洞。写两个页面，一个是教师录入成绩页面，一个是学生查询成绩页面。教师录入成绩页面表单有三个字段，课程id，学生id，成绩。录入提交以后，httpserver调用sqlite库使用sql语句写入数据库。然后是学生查询成绩表单，学生输入学生id，课程id，httpserver使用sql语句查询成绩后返回给用户。这里不需要做登录功能，课程也用直接输入id而不是下拉菜单的方式，或者其他选择的方式，而是直接输入id。为了体验最原始的web的开发过程。

大家用原始的方式写完了以后，后面我们再讲先进的方式如何写。以及原始的方法有什么问题，先进的方法如何规避了这些问题。