使用Django这种框架编程，第一步是定义模型，Django会自动把定义好的模型转化为数据库表结构。这种方式称为 ORM。所以我们在models里定义了三个类，然后执行了生成migrations和migrate。数据库的表就建立完成，以后每次修改了models，都需要执行makemigrations和migrate。才能将修改结果对应到数据库中。

下面，我们来写view。

views是Django功能实现应用功能的地方。如果你想写一个动态的页面，就在views中定义一个函数。这是最基本的方法。在此基本的方法上还可以有高级的，系统内置的一些高级的数据库增删改查的方法。最基本的views函数，是收到一个HttpRequest类型的参数，需要返回一个HTTPResponse类型的返回值。和http协议对应。在edu\_admin中的views.py写入一下内容。

首先，导入：

from django.http import HttpResponse

然后 写一个函数 ：

def index(request):

    return HttpResponse('<html><body>OK</body></html>')

这个函数就是一个基本的 “处理请求，返回响应”。

我们还需要把这个views，对应到一个路径上。也就是客户端如何调用这个views函数对应的功能。因为一个实用的web app只有一个views是不够的，可能有很多很多views。然后我们需要把这些views对应到不懂的url上。这样客户端才能访问。这个工作，是urls.py来完成的。（在 edu\_admin中建一个urls.py文件）

下面我们在urls.py中写如下内容。

from django.urls import path

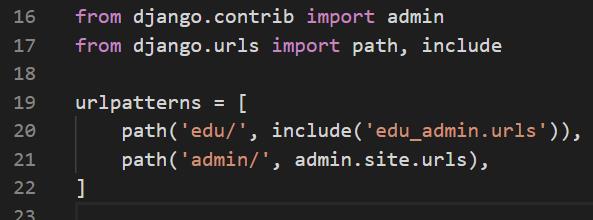
from .views import \*

urlpatterns = [

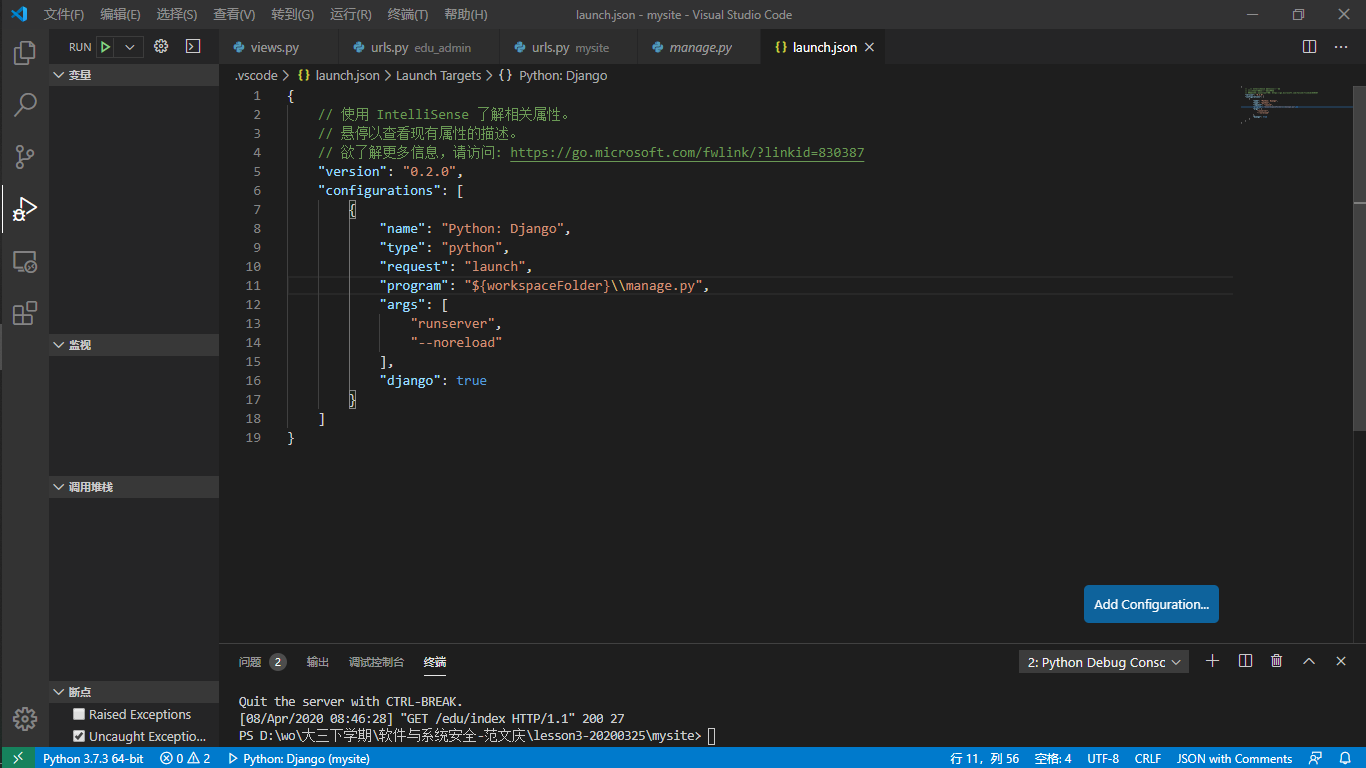
     path('index', index),

]

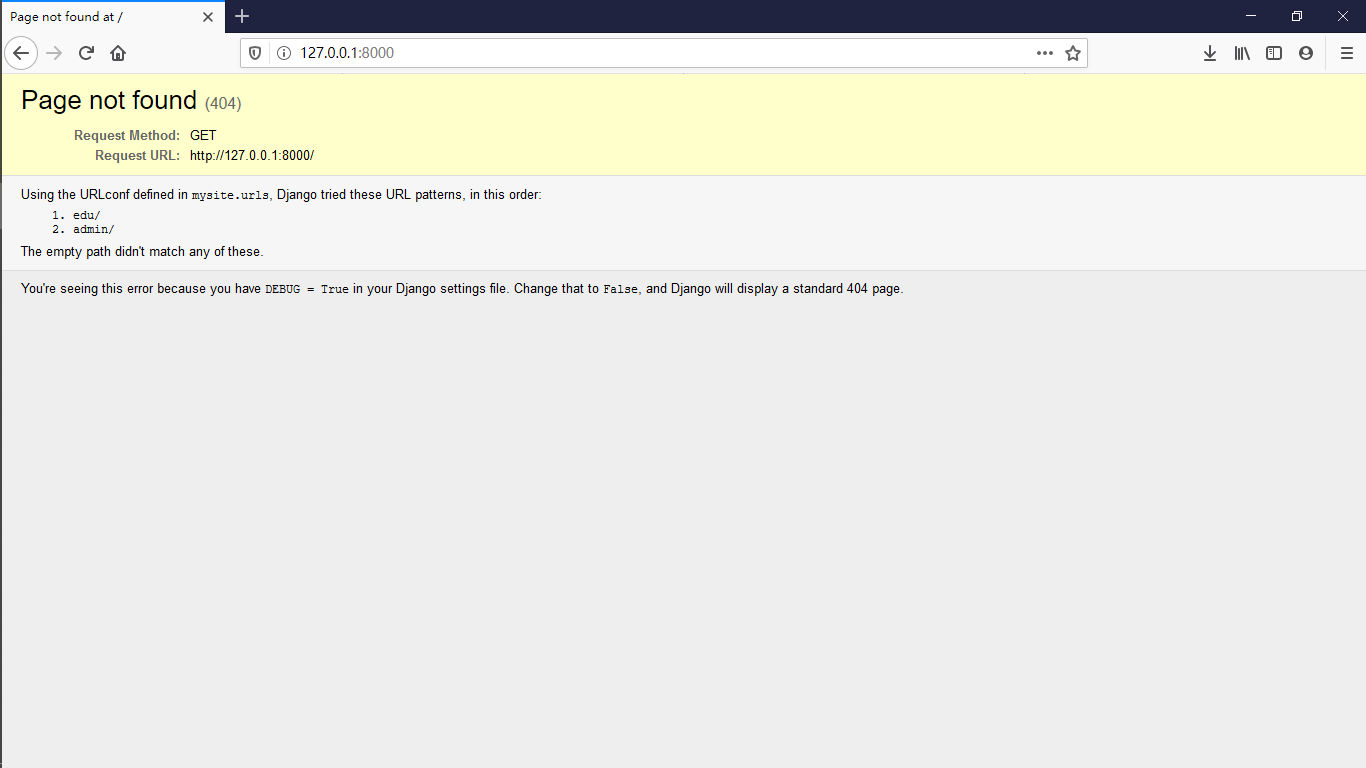
然后需要在 **主urls.py** ，也就是 mysite的urls.py中包括这个url配置。



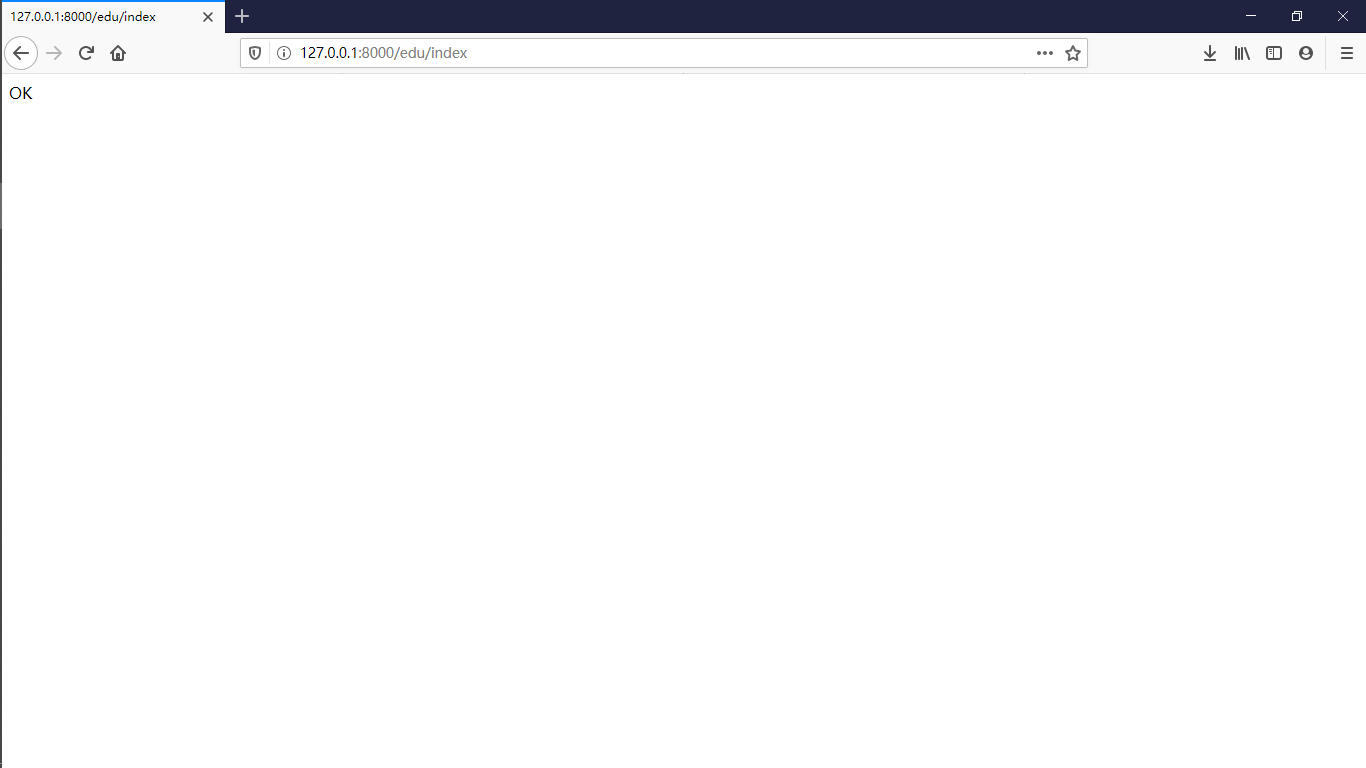
这是为了适应，可能有多个Django app共同工作的情况。比如我们这里的edu\_admin是一个app，polls又是一个app。有了以上修改，我们就可以运行我们的网站，看看效果了。运行是 python manage.py runserver 命令。也可以在vscode中调试运行，选择Django。生成的launch.json如下：



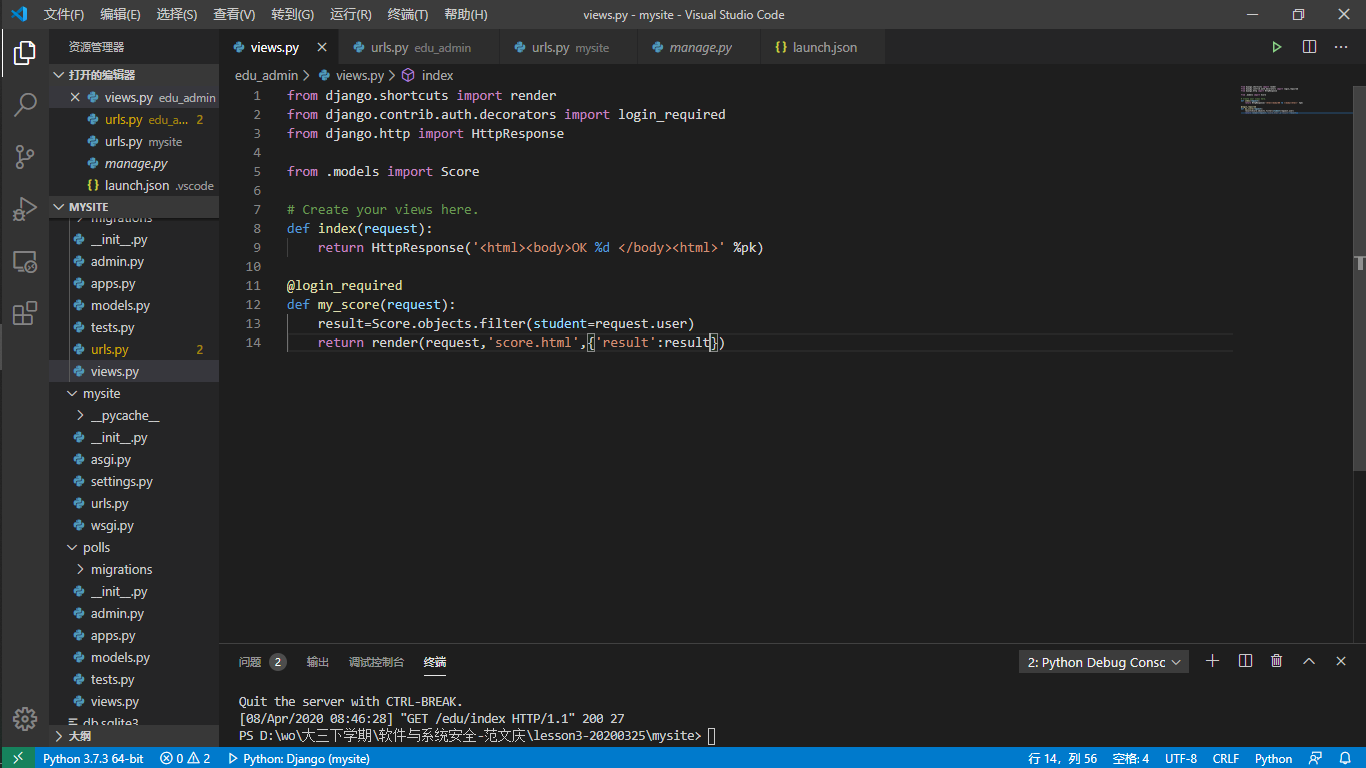
分别访问http://127.0.0.1:8000



和http://127.0.0.1:8000/edu/index



如果这一步有，就说明我们 urls和views配合工作成功。用户在浏览器中输入路径，django把这个url对应到一个views函数上。views函数处理HttpRequest。返回HttpResponse。这个工作流程跑通。把url对应到一个views函数的过程，专业术语叫“路由”，注意，这个路由不是路由器的那个路由。路由器的路由是IP层的IP数据包分发功能。web框架的路由只是借用了这个概念，引申为 web程序中url和处理函数的对应关系。以后我们要开发一个大型的网站，也是这么写。注意一个url是可以带参数的。views函数也是可以带参数的。比如：



像上图中红框中修改，即可有动态效果（如访问url为http://127.0.0.1:8000/edu/admin/100时，显示结果为 OK 100）其中的pk为参数，且只允许整数，路由系统会把这个参数赋值传递给views函数的pk参数。下一步，如何在views中访问数据库。这里我讲一下，如何在views函数中访问数据库。关键在第5行和第14行。先从models中导入模型类。然后调用这个模型类的objects的filter方法，就完成了一次sql select。filter函数的参数是就相当于查询的过滤条件。我们要查询的是 student为当前登录用户的Score表的记录。Score.objects.filter(student=request.user)。这条命令即可。Django中，当前登录用户，就在request对象的user属性中。views写了还不够。我们还需要修改一下模型。Django是有默认的用户管理系统的。用户相关的数据库表结构其实Django已经建立好了。但是我们这里用了student表来作为系统用户。所以我们要告诉Django不要用系统默认的用户模型了，用Student。

首先在 models.py中 导入：

from django.contrib.auth.models import AbstractUser

这个是Django默认的用户管理的数据库模型，然后对其进行继承修改。AbstractUser已经有很多数据库字段了，比如密码肯定是需要以某种方式保存到数据库中的。这些字段 AbstractUser都有了。我们在AbstractUser的基础上，扩充几个我们要用的字段就可以了。Student继承了AbstractUser后。告诉Django用Student作为系统用户管理的数据库模型。在mysite settings.py也就是整个站点的配置文件中，增加一条：

AUTH\_USER\_MODEL = 'edu\_admin.Student'

告诉Django，使用 edu\_admin 的Student作为用户管理登录授权的模型。代码修改完以后。这里是不是涉及到数据库修改啊。所有要进行数据库表结构的migrate。

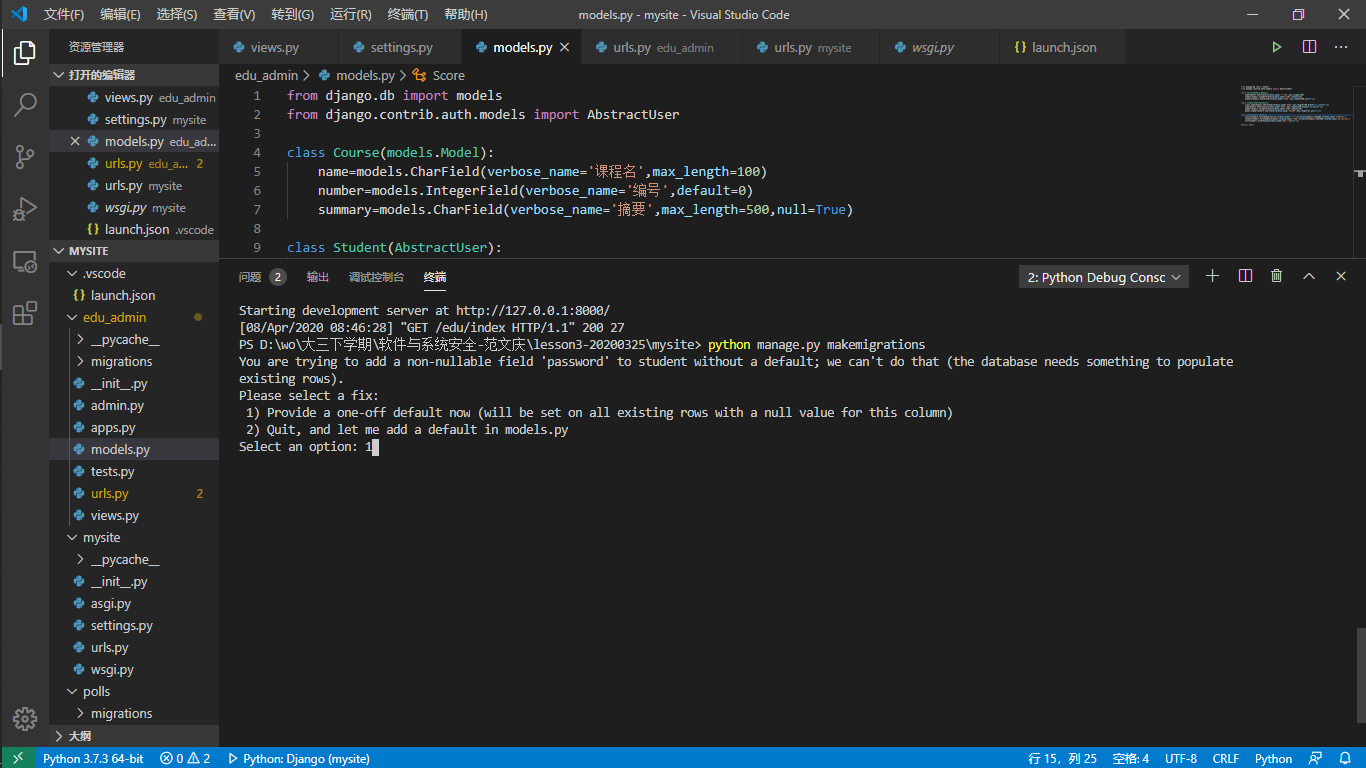
python manage.py makemigrations

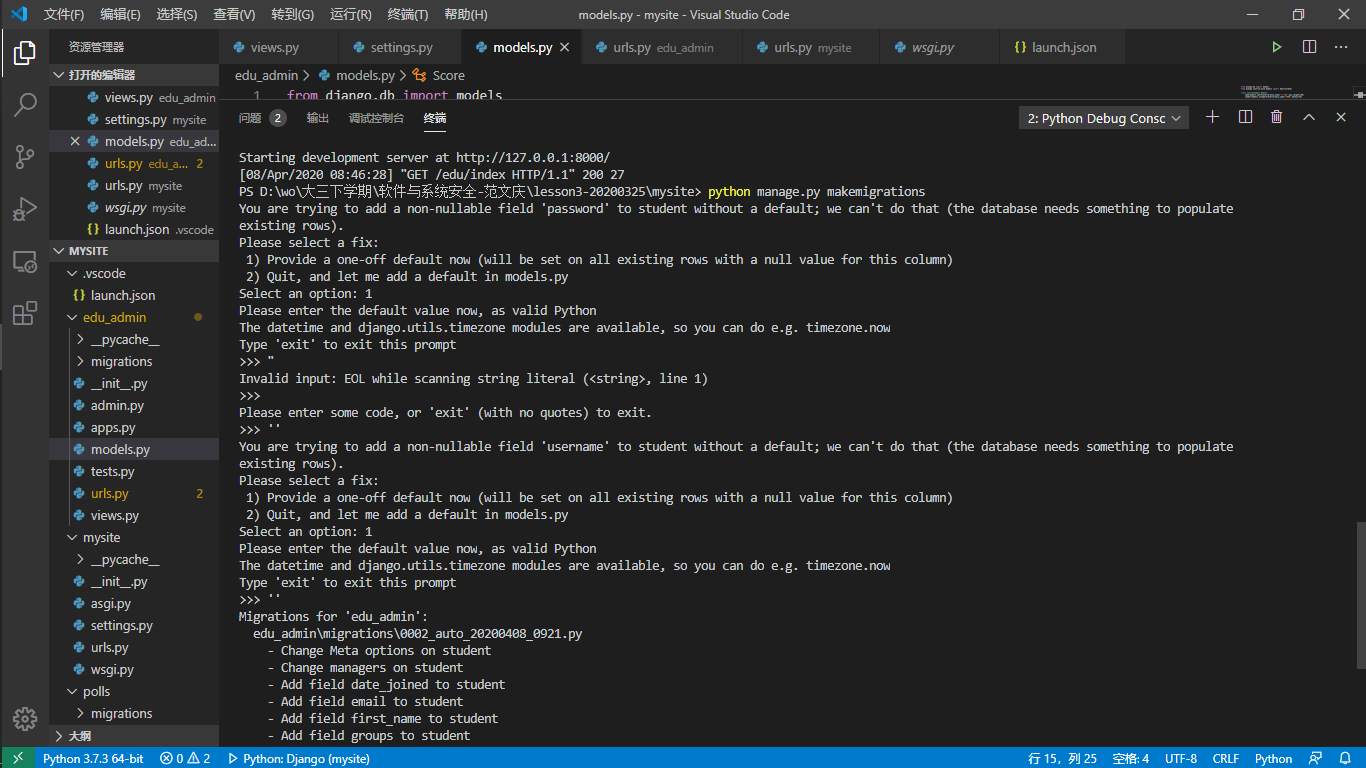
python manage.py migrate

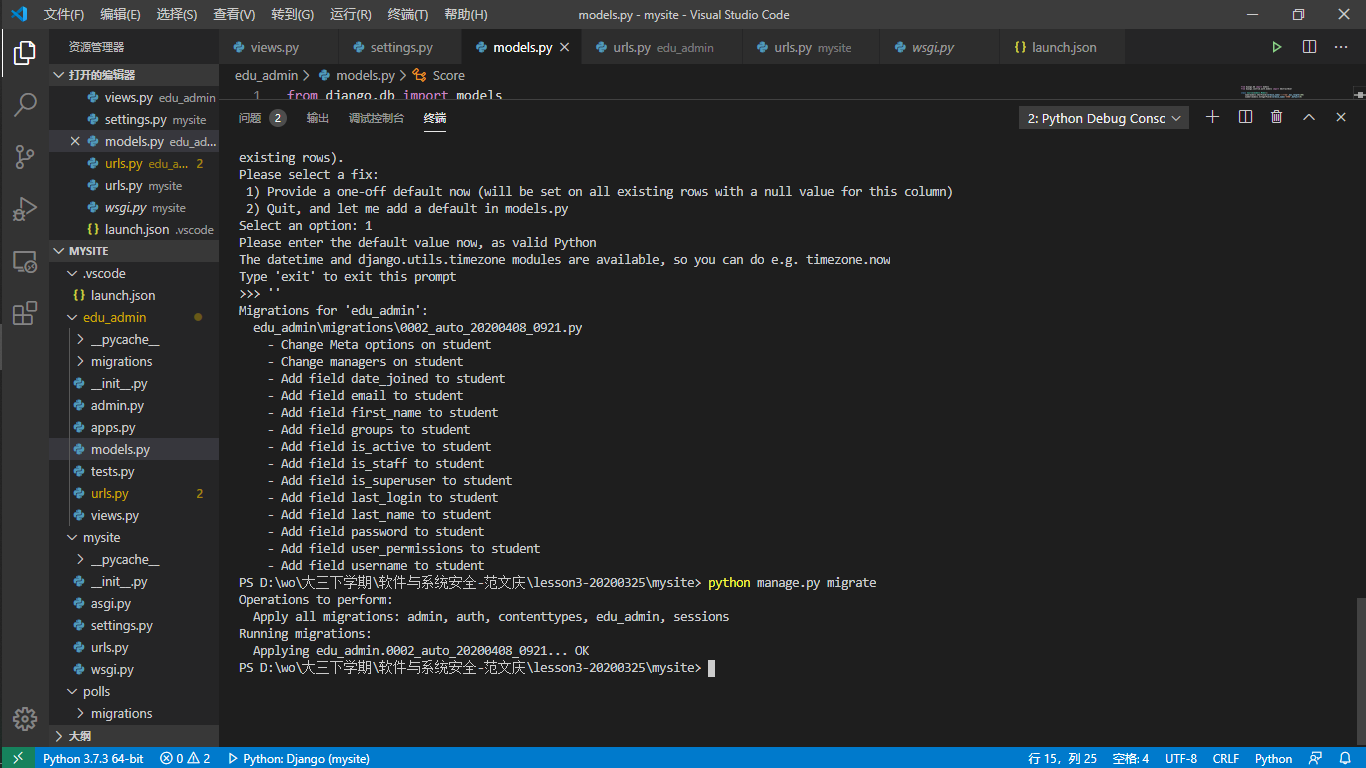
前一步生成迁移文件，后一步实施迁移。

迁移过程中，可能会要求输入一些默认值。

全部都输入为空字符串就可以 ''，对默认值选1，然后输入’’空字符串。（两个单引号）

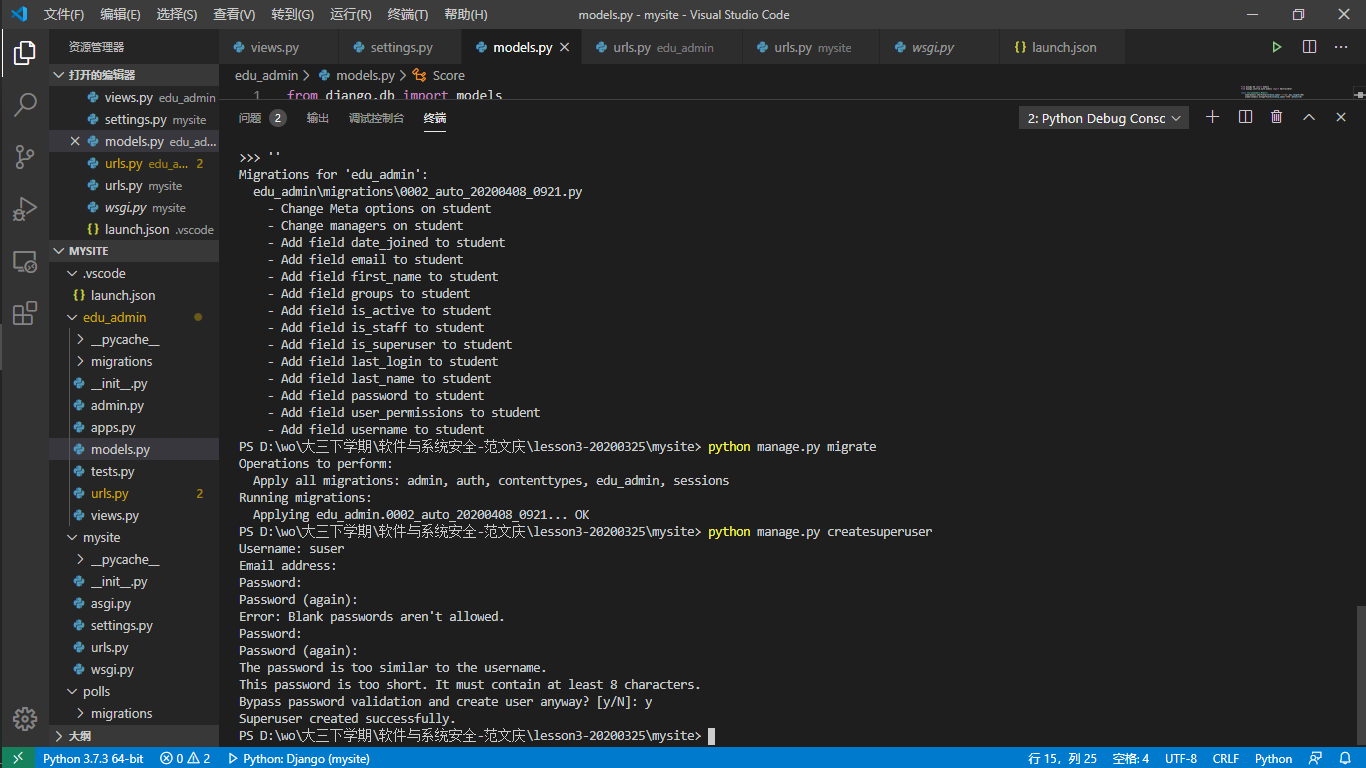






下面，我们尝试在数据库中写入一些数据。然后测试看看Django的数据库访问是否正常。最原始的方法就是在sqlite.exe 中用sql语句插入。但是这个方法容易把数据搞乱了。而且比如用户密码这样的东西，不是明文保存的。所有插入会有问题。我们用Django的方式，先建立一个超级用户：

python manage.py createsuperuser



（这里我设置的用户名和密码均为suser）

建立一个管理员账户。建立好了以后，用sqlite3.exe可以看到Student表多了一条记录。然后我们可以用Django的admin功能，用超级管理员录入数据。Django的admin相当于一个数据管理的超级权限后台。可以直接操作数据库。在admin.py中录入以下内容：

from django.contrib import admin

# Register your models here.

from .models import Student, Course, Score

class ScoreAdmin ( admin.ModelAdmin ):

    list\_display = ('course', 'score', 'student')

admin.site.register(Student)

admin.site.register(Course)

admin.site.register(Score, ScoreAdmin)

这样直接就可以生成一个管理数据库的后台页面。访问 http://127.0.0.1:8000/admin/ 刚才新建的用户 登录后看到这个页面。可以录入一些课程，学生，和成绩了。