

说明文档

Contents

一、数据库安装与设计.....	1
1、MySQL 安装.....	1
(1) 检查系统是否已经安装 MySQL.....	1
(2) 安装 MySQL.....	1
2、数据库的设计.....	3
(1) MyUsr.....	3
(2) question_table.....	3
(3) 用户作答情况表.....	4
二、工程项目.....	5
1、Django 框架部署与安装。.....	5
(1) Anaconda 环境搭建.....	5
(2) 搭建 Django 框架.....	6
2、项目开发.....	7
(1) data 目录.....	7
(2) dealJson 目录.....	8
(3) django_question 目录.....	10
(4) question 目录.....	12
(3) manage.py.....	14
三、项目运行.....	15
1、在终端下运行.....	15
2、在 vim 下运行.....	15
(1) 对 Vim 进行配置.....	15
(2) 用 vim 打开项目.....	15
3、在 Pycharm 中运行.....	16
(1) pycharm 的配置.....	16
(2) pycharm 运行.....	18

此开发环境建立在 Ubuntu 18.04 系统下，使用 Django 和 MySQL 数据库搭建

一、数据库安装与设计

由于我对开源数据库 MySQL 的偏爱，此项目使用的是 MySQL 的数据库。

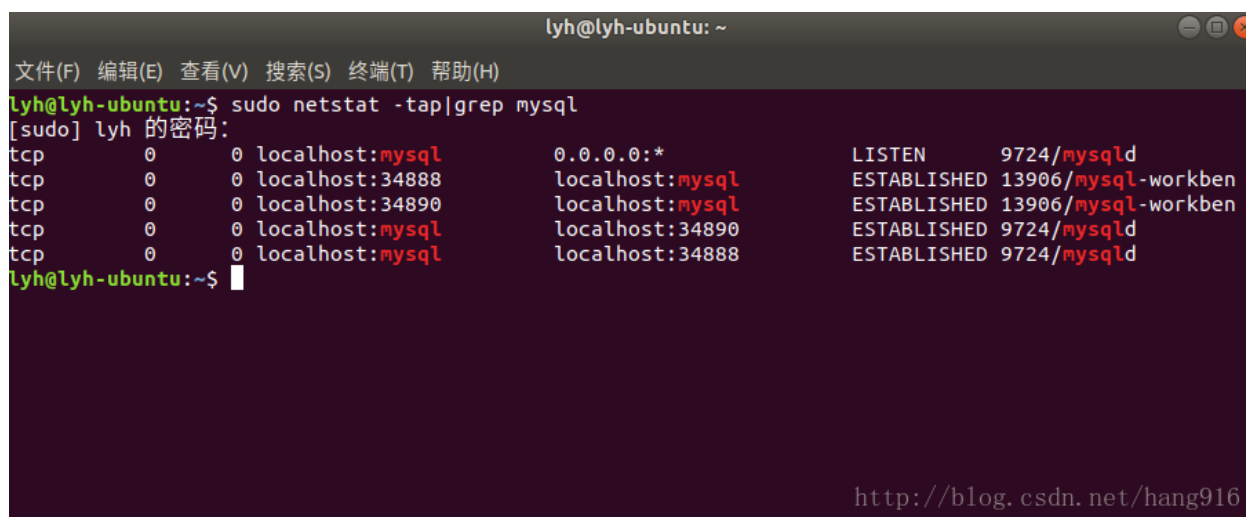
1、MySQL 安装

(1) 检查系统是否已经安装 MySQL

按下 Ctrl+Alt+T，打开终端，并在终端下输入

```
sudo netstat -tap|grep mysql
```

若如图 1-1 存在 mysql，则表示已经系统安装，跳过此过程，进入数据库设计阶段；若无任何显示，则代表系统未安装 MySQL，进入安装 MySQL 阶段。

A terminal window titled 'lyh@lyh-ubuntu: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'sudo netstat -tap|grep mysql' has been executed. The output shows several lines of network statistics for MySQL, including 'LISTEN 9724/mysql' and 'ESTABLISHED' connections. The prompt 'lyh@lyh-ubuntu:~\$' is visible at the bottom left. A URL 'http://blog.csdn.net/hang916' is at the bottom right.

```
lyh@lyh-ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
lyh@lyh-ubuntu:~$ sudo netstat -tap|grep mysql  
[sudo] lyh 的密码:  
tcp        0      0 localhost:mysql      0.0.0.0:*             LISTEN      9724/mysql  
tcp        0      0 localhost:34888      localhost:mysql      ESTABLISHED 13906/mysql-workben  
tcp        0      0 localhost:34890      localhost:mysql      ESTABLISHED 13906/mysql-workben  
tcp        0      0 localhost:mysql      localhost:34890      ESTABLISHED 9724/mysql  
tcp        0      0 localhost:mysql      localhost:34888      ESTABLISHED 9724/mysql  
lyh@lyh-ubuntu:~$  
  
http://blog.csdn.net/hang916
```

图 1-1 查看是否安装 MySQL

(2) 安装 MySQL

在终端内输入

```
sudo apt install mysql-server mysql-client
```

如图 1-2 所示，安装 MySQL

```
lyh@lyh-ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
lyh@lyh-ubuntu:~$ sudo netstat -tap|grep mysql  
lyh@lyh-ubuntu:~$ sudo apt install mysql-server mysql-client  
正在读取软件包列表... 完成  
正在分析软件包的依赖关系树  
正在读取状态信息... 完成  
将会同时安装下列软件：  
  libaio1 libevent-core-2.1-6 mysql-client-5.7 mysql-client-core-5.7 mysql-server-5.7  
  mysql-server-core-5.7  
建议安装：  
  tinyca  
下列【新】软件包将被安装：  
  libaio1 libevent-core-2.1-6 mysql-client mysql-client-5.7 mysql-client-core-5.7 mysql-server  
  mysql-server-5.7 mysql-server-core-5.7  
升级了 0 个软件包，新安装了 8 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级。  
需要下载 0 B/20.4 MB 的归档。  
解压缩后会消耗 160 MB 的额外空间。  
您希望继续执行吗？ [Y/n] y  
正在预设软件包 ...  
正在选中未选择的软件包 libaio1:amd64。  
(正在读取数据库 ... 系统当前共安装有 178198 个文件和目录。)  
正准备解包 .../0-libaio1_0.3.110-4_amd64.deb ...  
正在解包 libaio1:amd64 (0.3.110-4) ...  
^[[A正在选中未选择的软件包 mysql-client-core-5.7。  
正准备解包 .../1-mysql-client-core-5.7_5.7.21-0ubuntu0.17.10.1_amd64.deb ...  
http://blog.csdn.net/hang916
```

图 1-2 安装 MySQL

期间会提示你输入 root 用户（MySQL 的用户）密码，如图 1-3 所示，会提示你输入两遍，输入并按回车键确认。

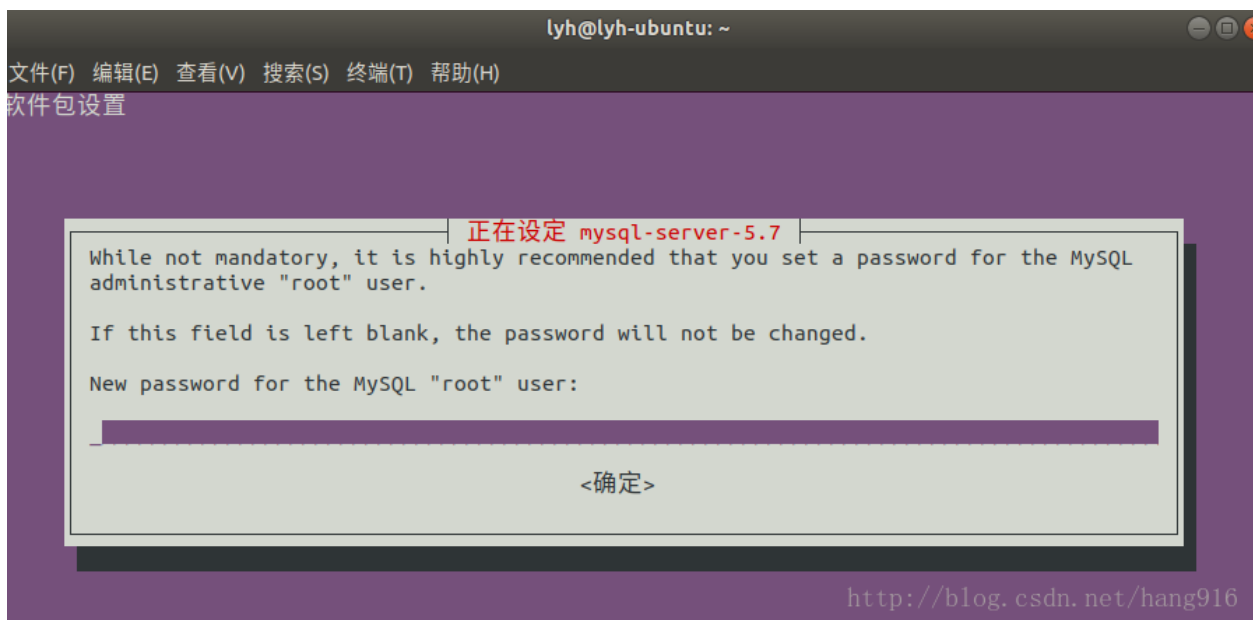


图 1-3 设置密码

安装完成后可以用（1）过程进行测试安装完成与否

sudo netstat -tap|grep mysql

2、数据库的设计

在终端下输入 `mysql -u root -p`，进入数据库。

创建数据库，并且指定编码

```
CREATE DATABASE `django_question` CHARACTER SET utf8 COLLATE utf8_general_ci;
```

数据库表的设计

数据库中主要存放三个表：用户表（MyUser），题目表（question_table），用户作答情况表（user_answer）。

（1）MyUsr

用户表一共有 6 个字段，如图 1-4 所示。

```
mysql> select * from MyUser;
+----+-----+-----+-----+-----+-----+
| id | username          | password | email          | sex  | gender |
+----+-----+-----+-----+-----+-----+
| 1  | MyUser object (None) | 123456   | asdfas@ssl.com | male |        |
| 4  | adfgasd           | 213456   | sd@como        | male | first  |
| 6  | admin             | yh960916 | 123213213@asdfas | female | third  |
| 8  | 123                | 123      | s1111@123m     | male | second |
+----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

图 1-4 MyUse 表

id：自增字段（用户表的主键）。

username：用户名（唯一）。

password：用户密码。

email：用户注册邮箱（唯一）。

sex：用户性别。

gender：用户年级（高中一年级、二年级、三年级）。

（2）question_table

问题表有 6 个字段，如图 1-5 所示。

id：问题表主键，6 位字符串，前三位代表问题类型，后三位代表每个类型中的序号。

quest：问题。

answer：参考答案。

question_type：问题类型。

```
mysql> SHOW FULL COLUMNS FROM question_table;
```

Field	Type	Collation	Null	Key
id	varchar(20)	utf8_general_ci	NO	PRI
quest	varchar(300)	utf8_general_ci	NO	
answer	varchar(300)	utf8_general_ci	NO	
question_type	int(11)	NULL	NO	
img_location	varchar(300)	utf8_general_ci	NO	
answer_keyword	varchar(300)	utf8_general_ci	NO	

图 1-5 问题表

img_location: 问题图所在的位置。

answer_keyword: 答案的关键词。

(3) 用户作答情况表

用户作答情况表共有 5 个字段，如图 1-6 所示。

```
mysql> SHOW FULL COLUMNS FROM user_answer;
```

Field	Type	Collation	Null	Key	Default	Extra
id	int(11)	NULL	NO	PRI	NULL	auto_increment
version	int(11)	NULL	NO		NULL	
user_answer	varchar(300)	utf8_general_ci	NO		NULL	
question_id	varchar(20)	utf8_general_ci	NO	MUL	NULL	
username_id	varchar(40)	utf8_general_ci	NO	MUL	NULL	

图 1-6 用户作答情况表

id: 自增字段，用户作答情况表主键。

version: 用户对于此题目的第几次作答。

user_answer: 用户此次作答的答案。

question_id: 题目表中的 id。

username_id: 用户表中的用户名。

用户作答情况表，由 username_id, question_id, version 三个字段来确定唯一记录。

二、工程项目

此项目由 Django 框架搭建而成。

1、Django 框架部署与安装。

如果虚环境搭建完成跳过此步骤。

(1) Anaconda 环境搭建

Anaconda 是专注于数据分析的 Python 发行版本，包含了 conda、Python 等 190 多个科学包及其依赖项。支持 Linux, Mac, Windows 系统，提供了包管理与环境管理的功能，可以很方便地解决多版本 python 并存、切换以及各种第三方包安装问题。

先解释下 conda、anaconda 这些概念的差别。conda 可以理解为一个工具，也是一个可执行命令，其核心功能是包管理与环境管理。包管理与 pip 的使用类似，环境管理则允许用户方便地安装不同版本的 python 并可以快速切换。Anaconda 则是一个打包的集合，里面预装好了 conda、某个版本的 python、众多 packages、科学计算工具等等，所以也称为 Python 的一种发行版。

conda 将几乎所有的工具、第三方包都当做 package 对待，甚至包括 python 和 conda 自身！因此，conda 打破了包管理与环境管理的约束，能非常方便地安装各种版本 python、各种 package 并方便地切换。

在“下载[download]”文件夹下右击在终端打开，在终端依次输入

```
wget https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-x86_64.sh
```

```
ls -a
```

```
bash Anaconda*.sh
```

```
rm Anaconda*.sh
```

```
echo 'export PATH=~/.anaconda/bin:$PATH' >> ~/.bashrc
```

```
source ~/.bashrc
```

```
conda update conda
```

```
source ~/.anaconda/bin/activate root
```

稍微解释下【可以跳过】：

第一句，下载 Anaconda

第二句，查看是否存在 Anaconda

第三句，运行 Anaconda

第四句，删除 Anaconda

第五句，将 Anaconda 的环境变量添加到 ~/.bashrc 文件下

第六句，重新更新 .bashrc 文件

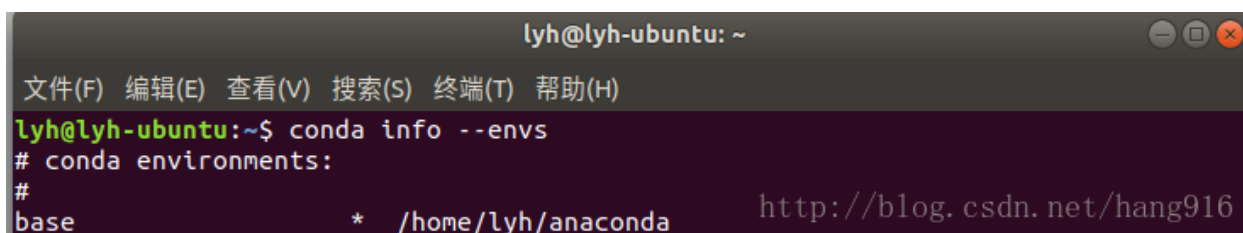
第七句，更新 conda

第八句，更新并执行 /anaconda/bin/activate 文件

(2) 搭建 Django 框架

在终端下输入下面命令查看环境配置。如图 2-1 所示：

```
conda info --envs
```



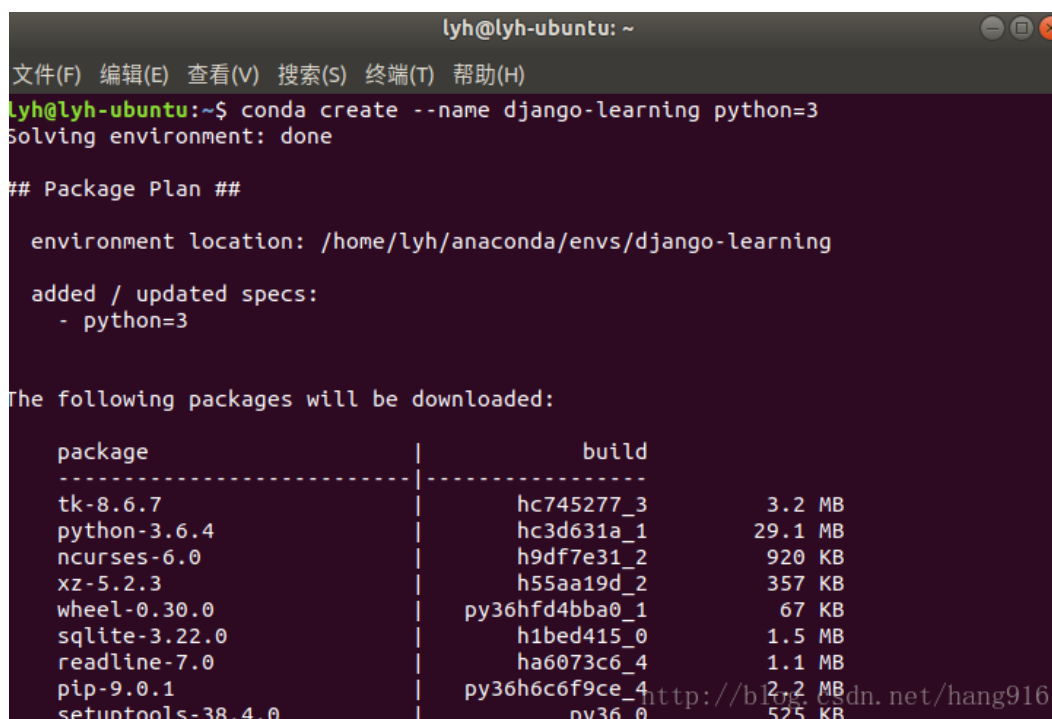
```
lyh@lyh-ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
lyh@lyh-ubuntu:~$ conda info --envs  
# conda environments:  
#  
base * /home/lyh/anaconda
```

图 2-1 查看环境配置

接下来创建一个新的环境来安装 Django，使用 "conda create" 命令来创建，如图 2-2 所示。

```
conda create --name django-learning python=3
```

也可以写成 "conda create -n django-learning python=3"



```
lyh@lyh-ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
lyh@lyh-ubuntu:~$ conda create --name django-learning python=3  
Solving environment: done  
  
## Package Plan ##  
  
environment location: /home/lyh/anaconda/envs/django-learning  
  
added / updated specs:  
- python=3  
  
The following packages will be downloaded:  
  
package | build | size  
-----|-----|-----  
tk-8.6.7 | hc745277_3 | 3.2 MB  
python-3.6.4 | hc3d631a_1 | 29.1 MB  
ncurses-6.0 | h9df7e31_2 | 920 KB  
xz-5.2.3 | h55aa19d_2 | 357 KB  
wheel-0.30.0 | py36hfd4bba0_1 | 67 KB  
sqlite-3.22.0 | h1bed415_0 | 1.5 MB  
readline-7.0 | ha6073c6_4 | 1.1 MB  
pip-9.0.1 | py36h6c6f9ce_4 | 2.2 MB  
setuptools-38.4.0 | py36_0 | 525 KB
```

图 2-2 配置虚环境

这句话的意思是创建了一个新的 Python 版本是 3 的，名字为 django-learning 的新环境，如图 2-2 所示。

接下来可以激活此环境。

```
source activate django-learning
```

安装完成后就可以进行安装 Django 了，使用 pip 进行安装

```
conda install django==2.0.4
```

使用下面语句进行测试自己的电脑安装好没有

```
python -c "import django;print(django.get_version())"
```

2、项目开发

项目目录结构如图 2-3 所示：

```
.. (up a dir)
</lyh/mycode/django_question/
> data/
> dealJson/
> django_question/
> question/
manage.py*
README.md
tree.md
```

图 2-3 项目目录结构

(1) data 目录

data 目录下存放的是 json 文件，如图 2-4 所示，这些 json 文件主要为了存放数据库中的数据。由于使用 Django 框架。Django 可以将命令【python manage.py loaddata *.json】将 json 文件存放到对应的表中。



图 2-4 数据库中的数据

experimentProblemJson.json：存放问题类型是 61（实验题）表的内容。

factorProblemJson.json：存放问题类型是 41（影响因素）表的内容。

outputProblemJson.json：存放问题类型是 21（产物）表的内容。

pigmentProblemJson.json：存放问题类型是 51（光合色素）表的内容。

rawoutputFunctionProblemJson.json：存放问题类型是 32（生物物质的作用）表的内容。

rawoutputRelationProblemJson.json：存放问题类型是 33（生物过程之间的关系）表的内容。

rawoutputWhatsProblemJson.json：存放问题类型是 31（生物过程中的原料和产物）表的内容。

rawPoblemJson.json：存放问题类型是 11（原料）表的内容。

用 rawProblemJson.json 举例说明文件存放的内容。如图 2-5 所示：

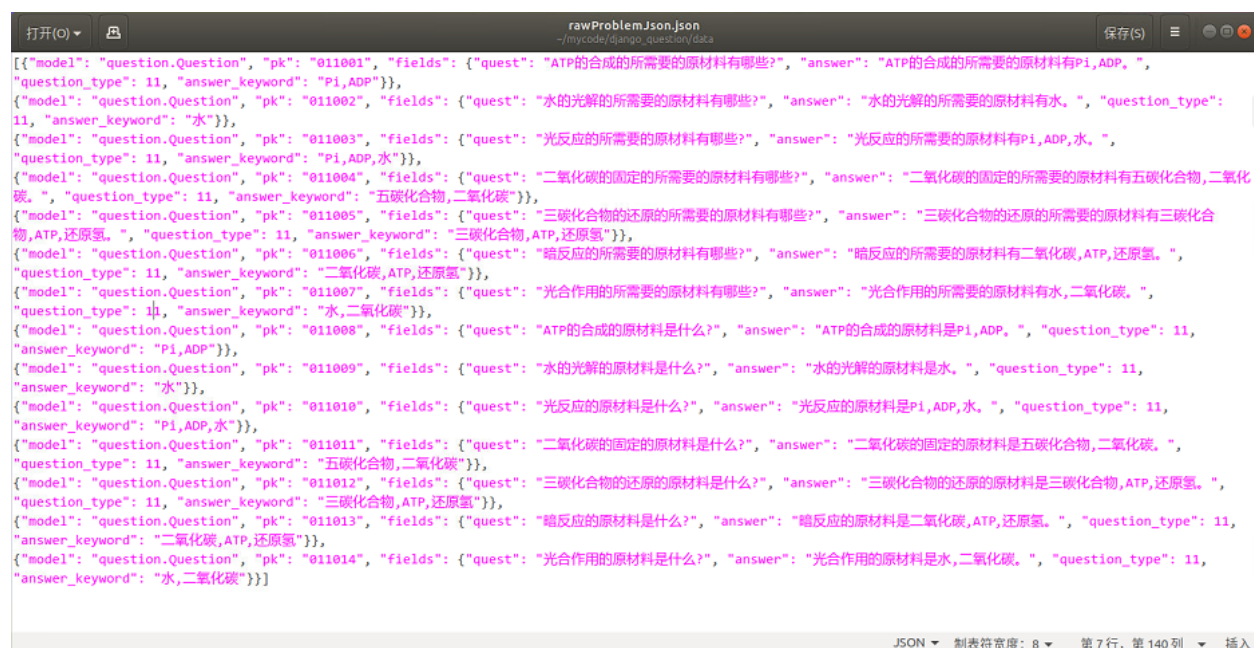


图 2-5 rawProblemJson.json

json 存放的是 array 类型数据。每一条数据是表中的一行。

【model】：代表使用数据库中哪一个表。此 json 对应了数据库中的 question_table 表

【pk】：对应表【question_table】中的关键词。

【field】：对应表【question_table】中的字段。

(2) dealJson 目录

将之前得到的问题文本写成上述对应的 json 文件。如图 2-6 所示：

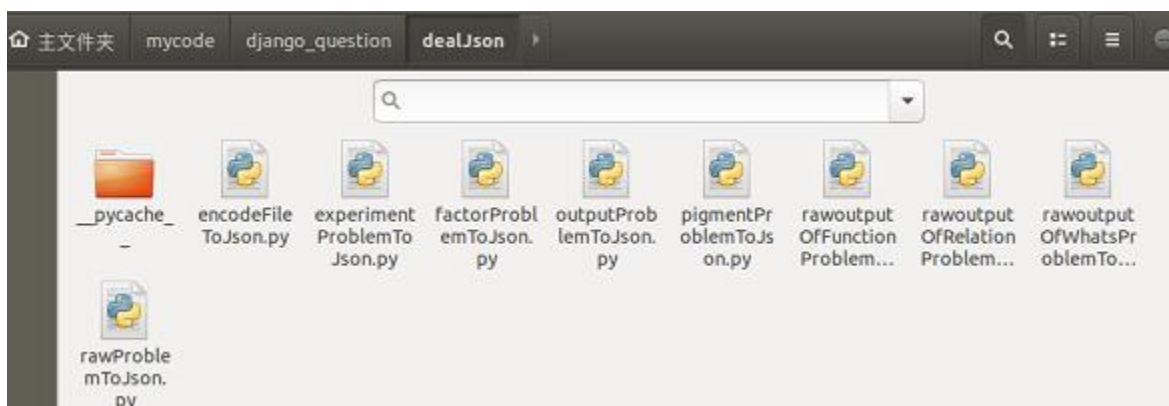


图 2-6 dealJson 目录

encodeFileToJson.py: 总的处理文件，封装了一些函数方便以后处理。如图 2-7 所示：

```
+--- 6 lines: def format_qa(self,content):-----
"""
    获得所有的问题和答案
"""
+--- 15 lines: def get_txt_data(self,file): 获取txt数据-----
"""
    将问题和答案进行匹配
"""
    # index:两个问题之间的行数差，如果非固定值，需手动调整下
+--- 16 lines: def trans_qa_to_pair(self,data,index):-----
+--- 15 lines: def write_json_data(self,content,question_type,pk_value,keyword=""):-----
+--- 6 lines: def write_json_file(self,file,data):-----
"""
    获取关键词
    算法流程：
    1、从文件中按行读取关键词
    2、判断是否有问题的关键词
    3、若有问题的关键词，只读取答案的关键词
    4、答案的关键词有几行
    5、将每道题答案的关键词添加到list
    6、为每道题的答案进行组合
    key_lines : 答案关键词有几行
    index : 一道题目问题及答案的关键词有几行
    is_quest : 是否有问题的关键词
"""
+--- 33 lines: def get_keyword(self, file, key_lines=1, index=1, is_question=True, delimiter=" "):--
"""
    input_file:输入文件
    output_file:输出文件
    question_type:问题类型（1-6,分别代表原来，产物，原料和产物，因素，光和色素，实验）
"""
+--- 30 lines: def topfuction(self,input_file,output_file,question_type,index,keywords=[]):-----
```

图 2-7 encodeFileToJson.py

注解：函数略解

- (a) `get_txt_data()`: 主要从 txt 问题文本中将数据写入到内存。
- (b) `trans_qa_to_pair()`: 将问题和答案写成配对函数。
- (c) `write_json_data()`: 将得到的数据写入的 json 字符串。
- (d) `write_json_file()`: 将 json 字符串写入 json 文件。
- (e) `get_keyword()`: 得到问题的关键词。
- (f) `topfunction()`: 整理上述函数。

`experimentProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/experiment/Problem.txt` , 生成 data 目录中的 `experimentProblemJson.json` 文件。

`factorProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/factor/FactorProblem.txt` , 生成 data 目录中的 `factorProblemJson.json` 文件。

`outputProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/output/problem.txt` , 生成 data 目录中的 `outputProblemJson.json` 文件。

`pigmentProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/pigment/PigmentProblem.txt` , 生成 data 目录中的 `pigmentProblemJson.json` 文件。

`rawoutputFunctionProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/raw_output/function/FunctionProblem.txt` , 生成 data 目录中的 `rawoutputFunctionProblemJson.json` 文件。

`rawoutputRelationProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/raw_output/relation/problem.txt` , 生成 data 目录中的 `rawoutputRelationProblemJson.json` 文件。

`rawoutputWhatsProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/raw_output/whats/problem.txt` , 生成 data 目录中的 `rawoutputWhatsProblemJson.json` 文件。

`rawProblemToJson.py` : 将 `/home/lyh/java/eclipse-workspace/QuestionGeneration/src/data/raw/problem.txt` , 生成 data 目录中的 `rawProblemJson.json` 文件。

(3) `django_question` 目录

主要是一些项目的配置文件，项目的根文件。结构如图 2-8 所示：

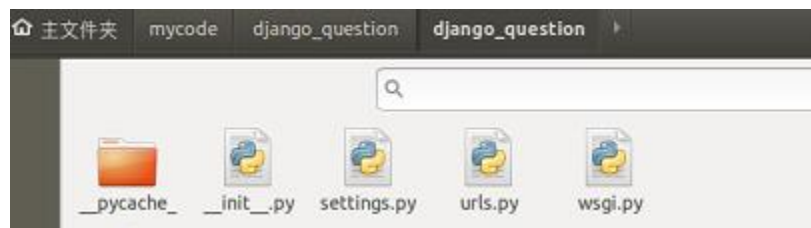


图 2-8 `django_question` 目录

init.py: 为了令 Django 项目是用 MySQL 数据库。

settings.py: 项目的一个设置文件。用于设置项目 APP 安装到项目中, 如图 2-9 所示, 设置项目 APP 的模板文件[TEMPLATES]的位置, 如图 2-10 所示; 设置项目 APP 静态文件[STATIC]的位置, 如图 2-11 所示; 项目数据库的配置, 如图 2-12 所示, 等等。

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'question',  
]
```

图 2-9 项目安装的 APP

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',  
                'django.template.context_processors.request',  
                'django.contrib.auth.context_processors.auth',  
                'django.contrib.messages.context_processors.messages',  
            ],  
        },  
    ],  
]
```

图 2-10 模板文件所在目录

```
STATIC_URL = '/static/'
```

图 2-11 静态文件配置

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'django_question',  
        'USER': 'root',  
        'PASSWORD': 'yh0916',  
        'HOST': '127.0.0.1',  
        'PORT': '3306'  
    }  
}
```

图 2-12 数据库配置

urls.py: 项目所在调用生成的根 URL。如图 2-13 所示。

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('question/', include('question.urls')),  
]
```

图 2-13 根 URL

此项目主要有两个地址，管理员地址和 question 【APP】 地址。

wsgi.py: 调用 django 接口。

(4) question 目录

此项目重点是 question 目录，也即 question 【APP】。目录结构如图 2-14 所示。

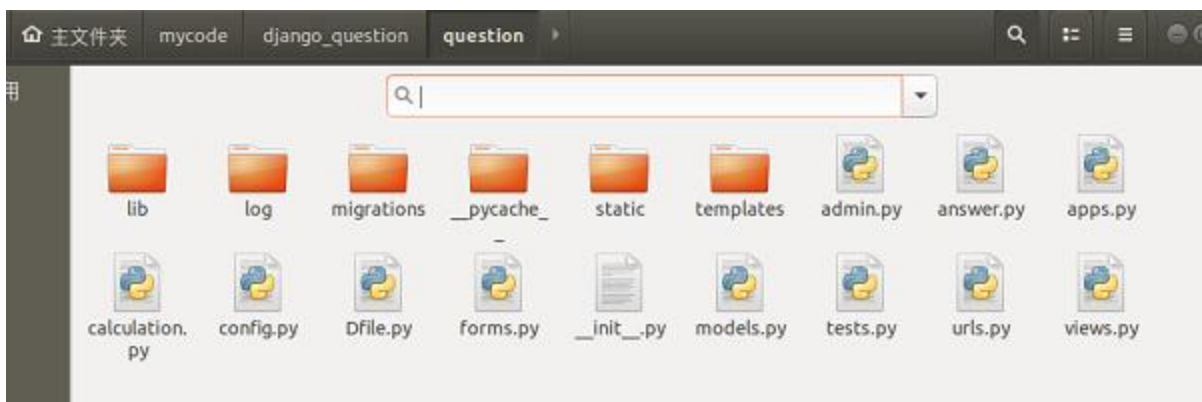


图 2-14 question APP

【lib】：里面存放的自己定义的 常量类，python 不含常量类型。故自己定义个常量类型方便以后可以产生一些常量类。

【log】：存放了一些文件，如图 2-15 所示。

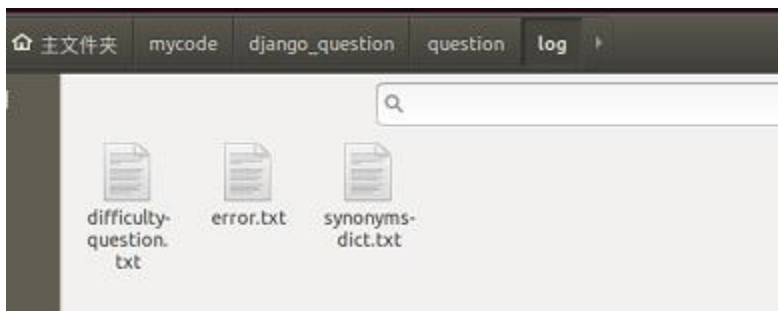


图 2-15 log 目录

(a) difficulty-question.txt: 问题类型设置难度系数的文件。

(b) synonyms-dict.txt: 同义词文件，每行是一个同义词词典。

(c) error.txt: 如果忘记写文件地址, 读取错误, 输出此文件。

【migrations】: 每次数据迁移时, django 生成的文件。

【static】: 存放此项目的静态文件, 例如 img、css、文件。

(a) img: 存放图片

(b) css: 存放所需要的层叠样式表

【templates】: 存放模板文件, django 根据模板文件动态生成网页, 目录结构如图 2-16 所示。

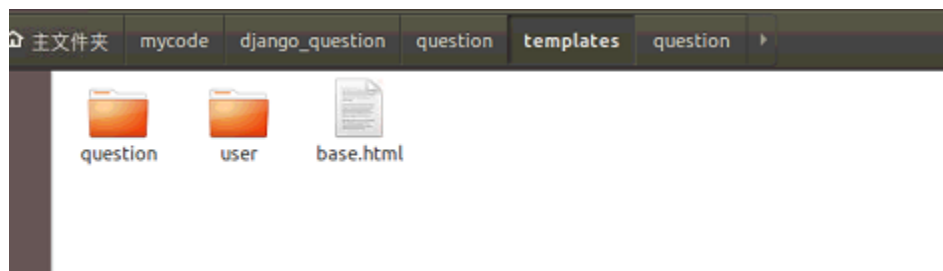


图 2-16 templates 目录结构

有两个目录, question 和 user, question 目录是存放做题和答题的模板文件, 如图 2-17 所示。user 存放的时登陆和注册的模板文件, 如图 2-18 所示。

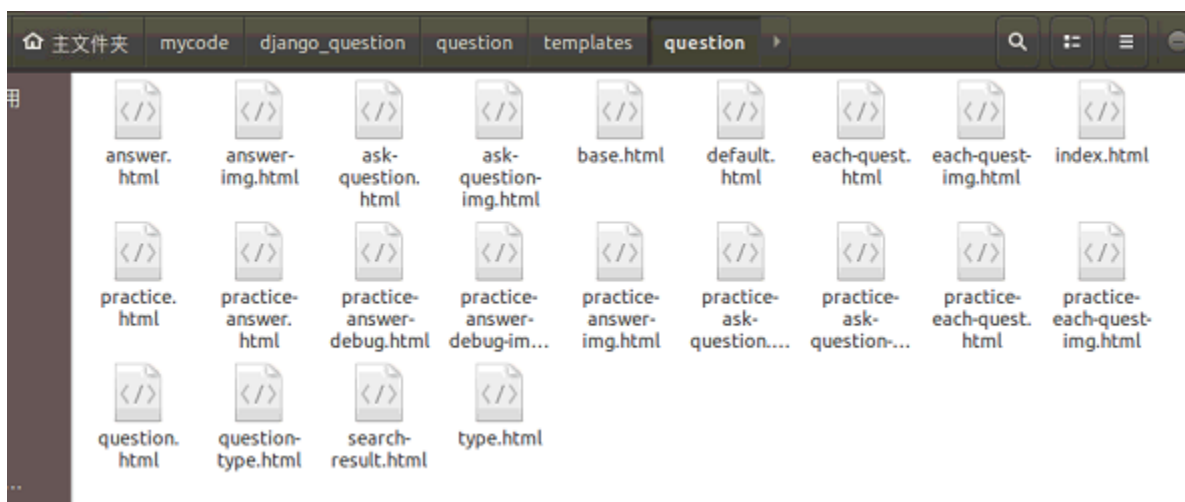


图 2-17 question 目录

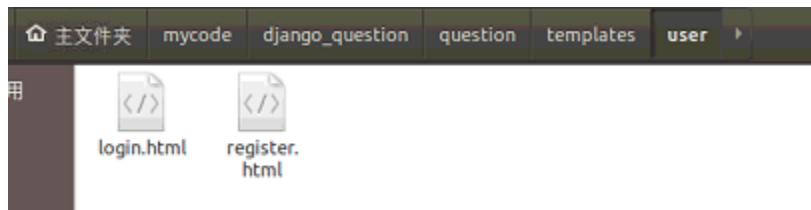


图 2-18 user 目录

【answer.py】：用户作答的类

【apps.py】：设置此 APP 的名字

【calculation.py】：用来获取用户作答的关键词、未作答的关键词、作答情况和参考答案的匹配度、以及从一系列问题中选取下一道问题。

【config.py】：设置一些控制变量，如图 2-19 所示



图 2-19 config.py

- (a) CORRECT_RATE: 学生的作答情况占参考答案的比率基于多少，基本可以视为学生作答正确。
- (b) DIFFICULTY_FILE: 题目难度系统所在的文件
- (c) SYNONYMS_DICT: 同义词词典所在的文件
- (d) DEBUG: 是否开启调试
- (e) STRATEGY: 选择一种出题策略

【Dfile.py】：用来处理 txt 文本，“#”做在的行为注释

【forms.py】：登陆和注册填写的表单

【models.py】：用来自定义数据库中的表

【urls.py】：生成的链接路径

【views.py】：对于每一个连接，操作对应的函数

(3) manage.py

操作 Django 的命令。例如：

- (a) python manage.py loaddata *.json # 将 json 文件写入对应的数据库
- (b) python manage.py makemigrations # 将数据库生成迁移文件
- (c) python manage.py migrate # 对数据库进行迁移
- (d) python manage.py startapp # 生成一个新的 web APP
- (e) python manage.py startproject # 生成一个新的项目
- (f) python manage.py test # 对项目进行单元测试
- (g) python manage.py runserver # 运行服务器

三、项目运行

此项目运行有三种方式：

1、在终端下运行

- (1) 激活 Django 所在的虚拟环境（确保虚环境有 django 插件）

```
source activate django      # 我的虚环境名称是就是 django
```

- (2) 切换至工程项目文件，项目文件根目录

```
cd ~/mycode/django_question
```

- (3) 调用 Django 项目下的 manage.py

```
python manage.py runserver
```

- (4) 打开浏览器输入：<http://127.0.0.1:8000>

2、在 vim 下运行

- (1) 对 Vim 进行配置

vim 配置文件所在位置：<https://github.com/YiFraternity/vim.git>

- 【1】在终端中调用如下命令下载配置文件。

```
git clone https://github.com/YiFraternity/vim.git
```

- 【2】将下载的配置文件.vimrc 放在【~】目录下

```
cp vim/.vimrc ~
```

- 【3】配置 vim

- 1) Vundle 是 Vim bundle 的简称，使用 git 来管理 vim 插件，有了它，安装其它插件就方便很多。

```
git clone https://github.com/VundleVim/Vundle.vim.git ~/.vim/bundle/Vundle.vim
```

- 2) 在终端中用 vim 打开.vimrc

```
vim ~/.vimrc
```

输入【:PluginInstall】进行安装插件

- 3) 配置完成，退出

输入命令【:wq】

- (2) 用 vim 打开项目

- 【1】在终端下输入以下命令：


```
cd ~/mycode/django_question
```

```
vim
```

【2】在 vim 下输入 【:Drunserver】

3、在 Pycharm 中运行

(1) pycharm 的配置

依次打开 File->Settings->Project->Project Interpreter，如图 3-1 所示。打开红色位置，选择 add...，添加 Python 解释器，如图 3-2 所示。按照图 3-2 打开，选择虚拟环境下，存在 django 包的 python 解释器，如图 3-3 所示。接下来依次点击确定，如图 3-4 所示项目显示为绿色，配置完成。

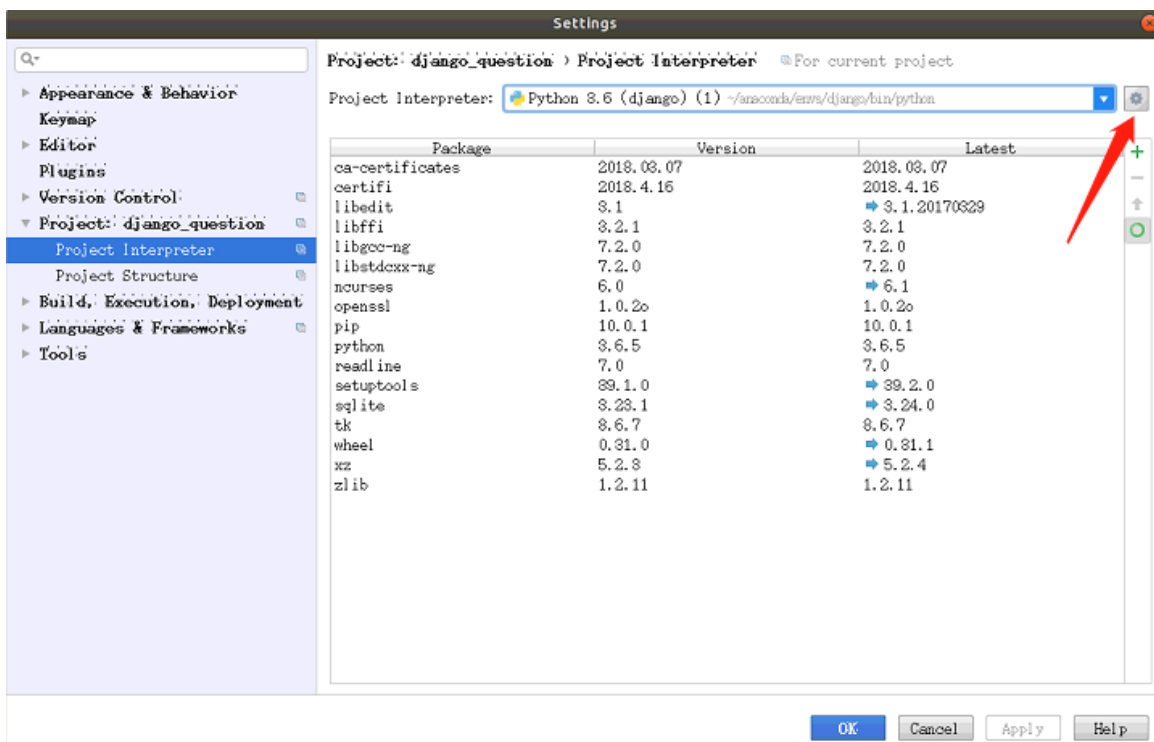


图 3-1 settings

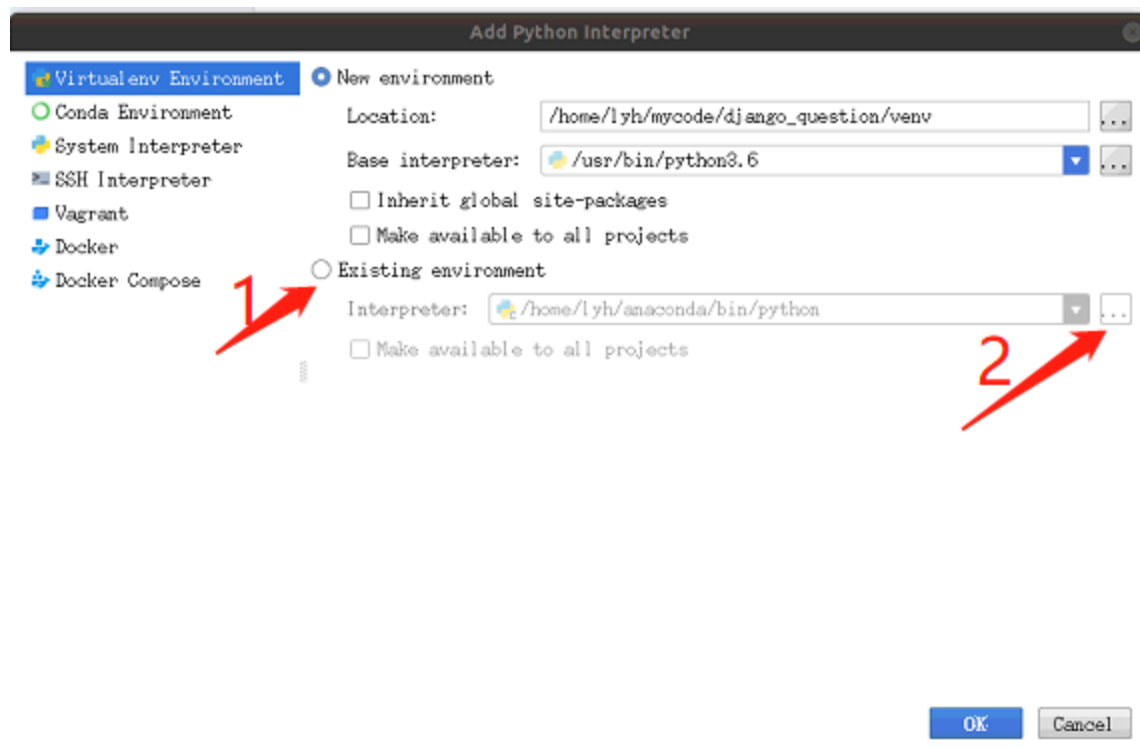


图 3-2 添加 python 解释器

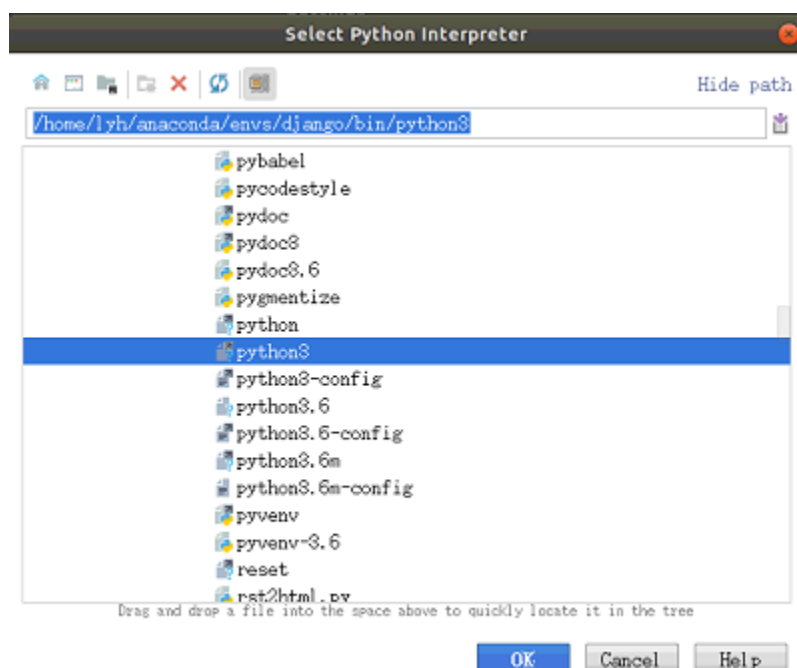


图 3-3 python 解释器所在的路径

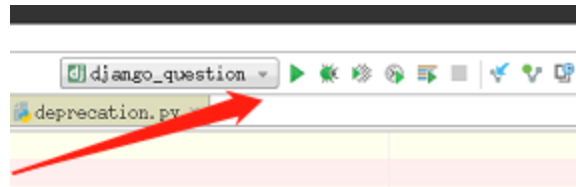


图 3-4 项目显示绿色

(2) pycharm 运行

按照图 3-5 所示，点击播放键就可以运行

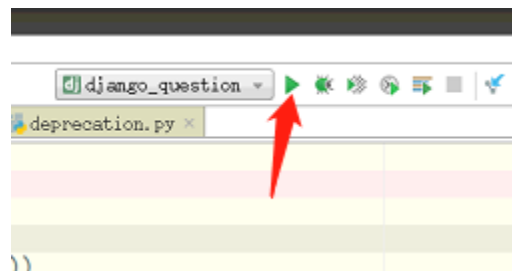


图 3-5 项目运行