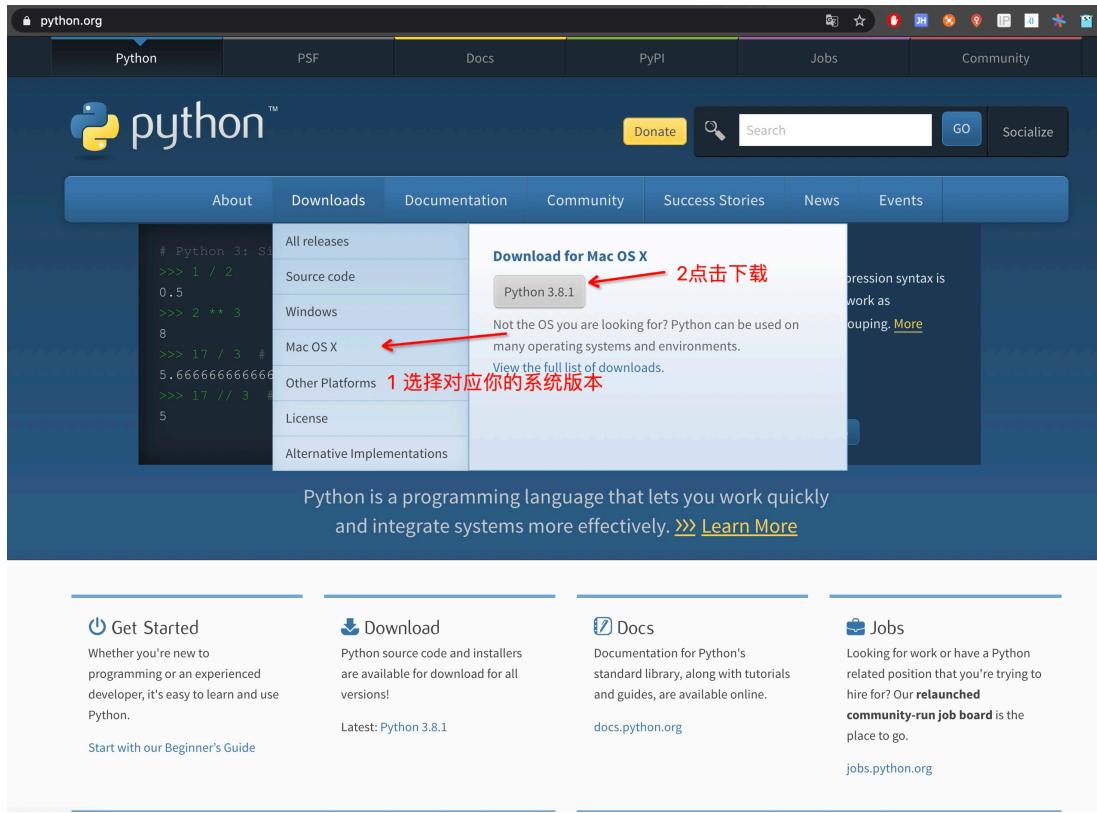
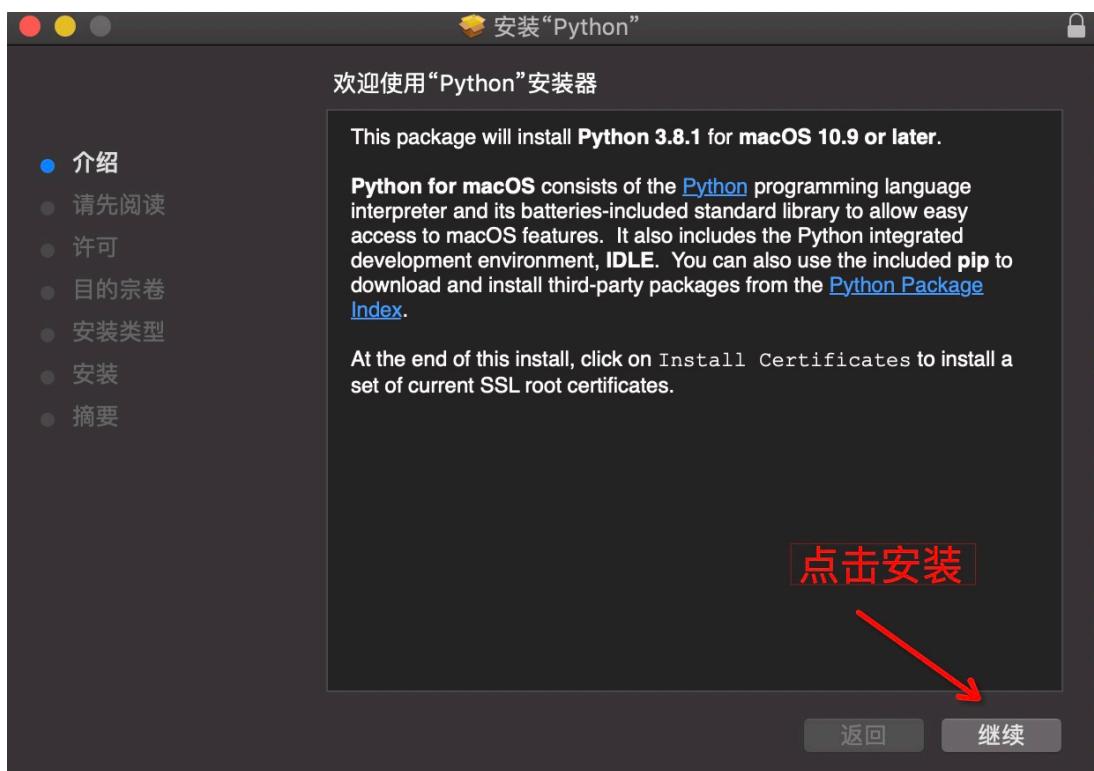


1.安装

网址: <https://www.python.org/>



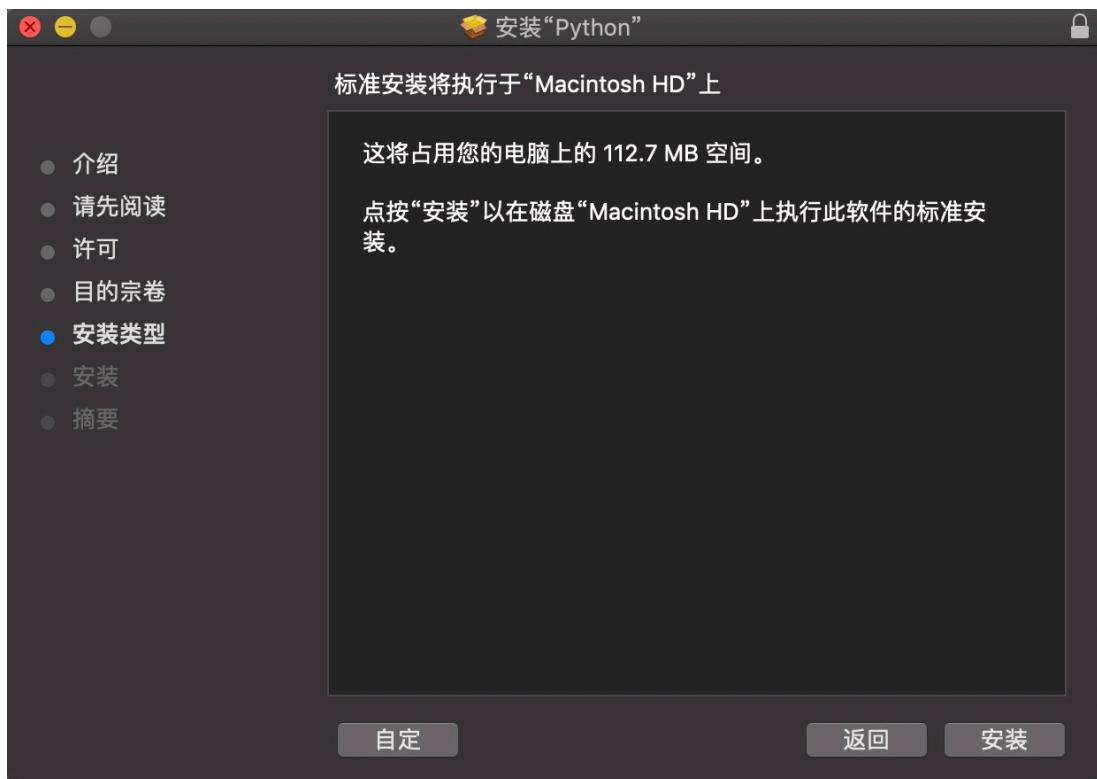
2.安装



3.一直点继续



4 记得勾选 添加Path路径



5. 安装完成

6. 打开pycharm, 如果没有, 请安装。 (其他编辑器也可以, 你开心就好)

网址: <https://www.jetbrains.com/>

The screenshot shows the JetBrains product page with a sidebar for filtering languages. The 'Python' checkbox is selected. The main area displays several IDEs: IntelliJ IDEA, PyCharm, Rider, CLion, DataLore, and DataGrip. A red arrow points from the 'PyCharm' section towards the 'Download' button.

Filters Reset X

LANGUAGES

- C/C++
- C#
- Dart
- DSL
- F#
- Go
- Groovy
- HTML
- Java
- JavaScript, TypeScript
- Kotlin
- Objective-C
- PHP
- Python
- Ruby
- Rust
- Scala
- SQL
- Swift
- VB.NET

IntelliJ IDEA 2019.3.1
The most intelligent JVM IDE
[Learn more](#) | [Buy](#)

PyCharm 2019.3.1
Python IDE for professional developers
[Learn more](#) | [Buy](#) DOWNLOAD

Rider 2019.3.1
Cross-platform .NET IDE
[Learn more](#) | [Buy](#)

CLion 2019.3.2
A smart cross-platform IDE for C and C++
[Learn more](#) | [Buy](#)

DataLore 1.0
Intelligent web application for data analysis
[Learn more](#)

DataGrip 2019.3.1
Many databases, one tool
[Learn more](#) | [Buy](#)

The screenshot shows the PyCharm download page. It features a Chinese service banner with the Chinese flag and the text '中文服务 支付宝'. The main heading is 'Download PyCharm'. It offers two download paths: 'Professional' (Windows, Mac, Linux) and 'Community' (Windows, Mac, Linux). A red arrow points from the 'Professional' section towards the 'Download' button.

PyCharm

What's New Features Learning Center Buy Download

中文服务 支付宝

JetBrains 致力于为开发者打造最高效智能的开发工具，总部以及主要研发中心位于欧洲的捷克。JetBrains针对个人开发者及企业组织提供不同的授权方式。我们提供多种付款方式，包含银联卡、各种国际信用卡、PayPal、支付宝和微信支付。企业户也可通过银行汇款的方式完成付款。若您有任何购买或授权上的疑问，欢迎联系我们的中文销售代表为您服务。
立即联系[中文销售代表](#)

The screenshot shows the PyCharm download page for the Professional edition. It includes a version history section for PyCharm 2019.3.1, system requirements, installation instructions, and other versions. The main download section has tabs for Windows, Mac, and Linux. It offers 'Professional' and 'Community' editions. A red arrow points from the 'Professional' section towards the 'Download' button. A callout box at the bottom right encourages using the Toolbox App for updates.

Download PyCharm

Windows Mac Linux

Professional
For both Scientific and Web Python development. With HTML, JS, and SQL support.
[Download](#) [Free trial](#)

Community
For pure Python development
[Download](#) [Free, open-source](#)

Get the Toolbox App to download PyCharm and its future updates with ease

安装pycharm 完成。

7.由于编辑器需要破解，才能长时间使用，当然也可以买正版。

地址：<https://www.jetbrains.com/store/?fromNavMenu#commercial?billing=yearly>

The screenshot shows the JetBrains website's product selection page. At the top, there are logos for China and Alipay, followed by text in Chinese: "JetBrains致力于为开发者打造最高效的智能工具, 总部以及主要研发中心位于欧洲的捷克。我们提供多种付款方式, 包括银联卡、各种国际信用卡, PayPal, 支付宝和微信支付。您有任何购买或授权上的疑问, 欢迎联系我们的中文销售代表为您服务。立即联系中文销售代表" and a close button "X". Below this, there are filter options: "适用" (Applicable), "组织" (Organization), "于个人使用的" (For individual use), "特别优惠" (Special offer), and "常见问题解答" (FAQ). Two tabs are shown: "每年帐单节省2个月" (Save 2 months on annual billing) and "每月帐单" (Monthly billing). The main content area displays three product packages:

- 所有产品包** (All Products): **美金\$ 649.00** / 用户 第一年. 美金\$ 519.00 / 2年
美金\$ 389.00 / 第三年起. Includes a "立即购买" (Buy Now) and "获得报价" (Get Quote) button.
- IntelliJ IDEA旗舰版** (IntelliJ IDEA Ultimate): **美金\$ 499.00** / 用户 第一年. 美金\$ 399.00 / 2年
美金\$ 299.00 / 第三年起. Includes a "立即购买" (Buy Now) and "获得报价" (Get Quote) button.
- ReSharper Ultimate + Rider**: **美金\$ 449.00** / 用户 第一年. 美金\$ 359.00 / 2年
美金\$ 269.00 / 第三年起. Includes a checked checkbox for "包括车手" (Includes Rider) and a "立即购买" (Buy Now) and "获得报价" (Get Quote) button.

8. Pycharm激活

在python交流群里看到的文章，我试了可以，就直接拿来复制，粘贴了。

1.点击链接

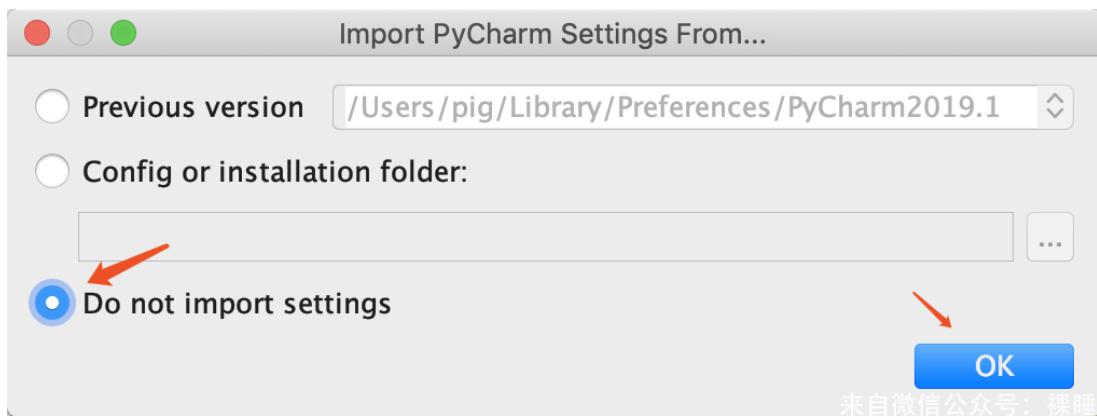
<https://pan.baidu.com/s/1a7kEHLr8anelPdSm0iquew> 下载补丁文件 jetbrains-agent.jar

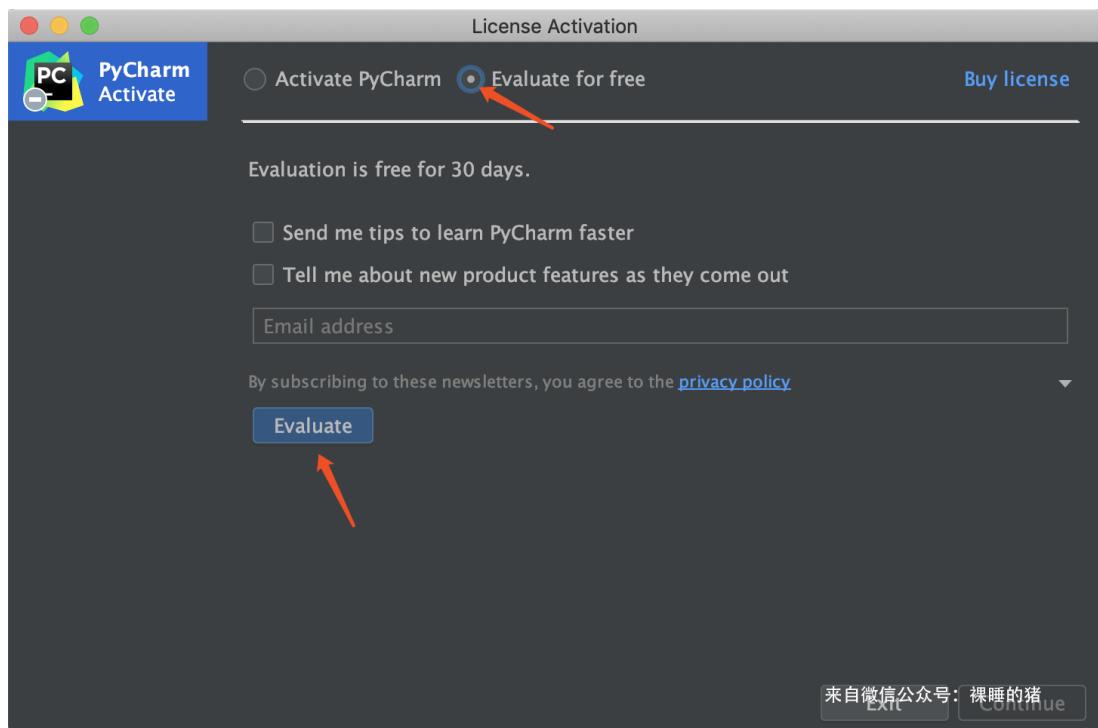
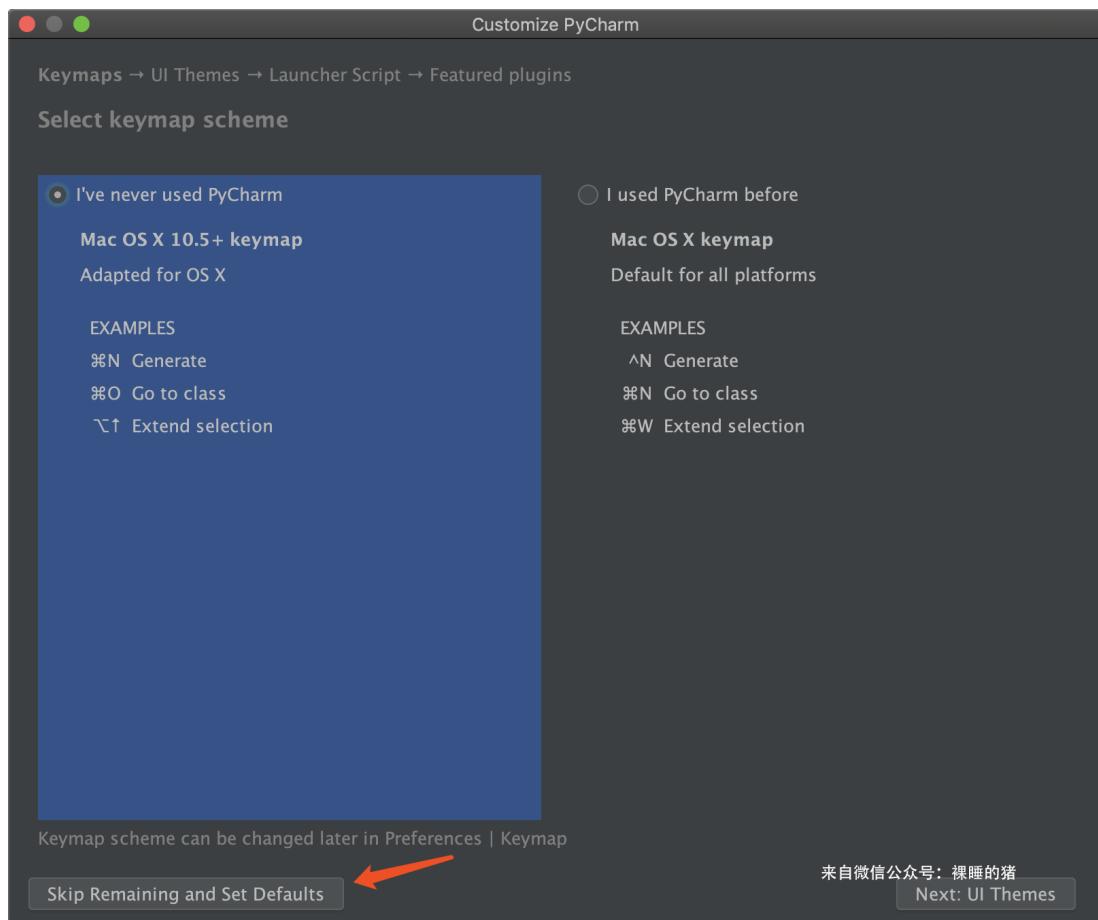
(并将它放置到 pycharm安装目录的\bin目录下 (位置可随意, 放这里是怕误操作删除了破解文件)。

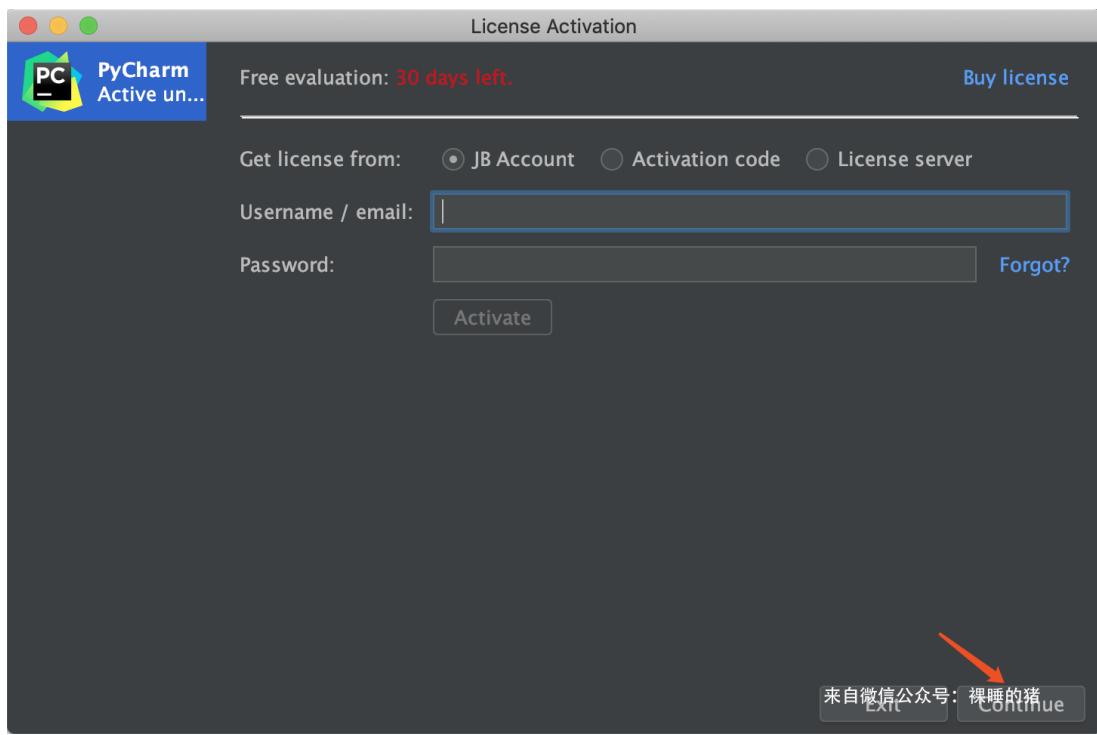
2.先点击试用



如果你是刚下载的pycharm，则需要点击激活窗口的“Evaluate for free”免费试用，然后再创建一个空项目，这样就可以进入到pycharm的工作页面。





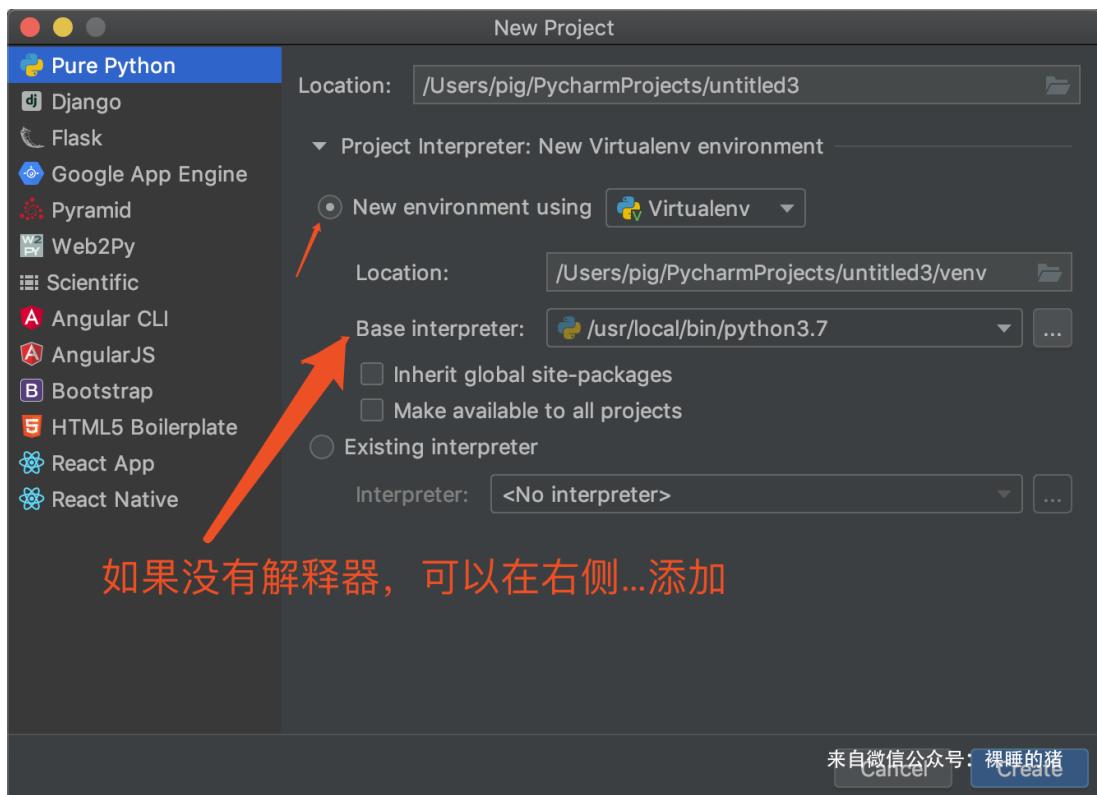


3.修改配置文件

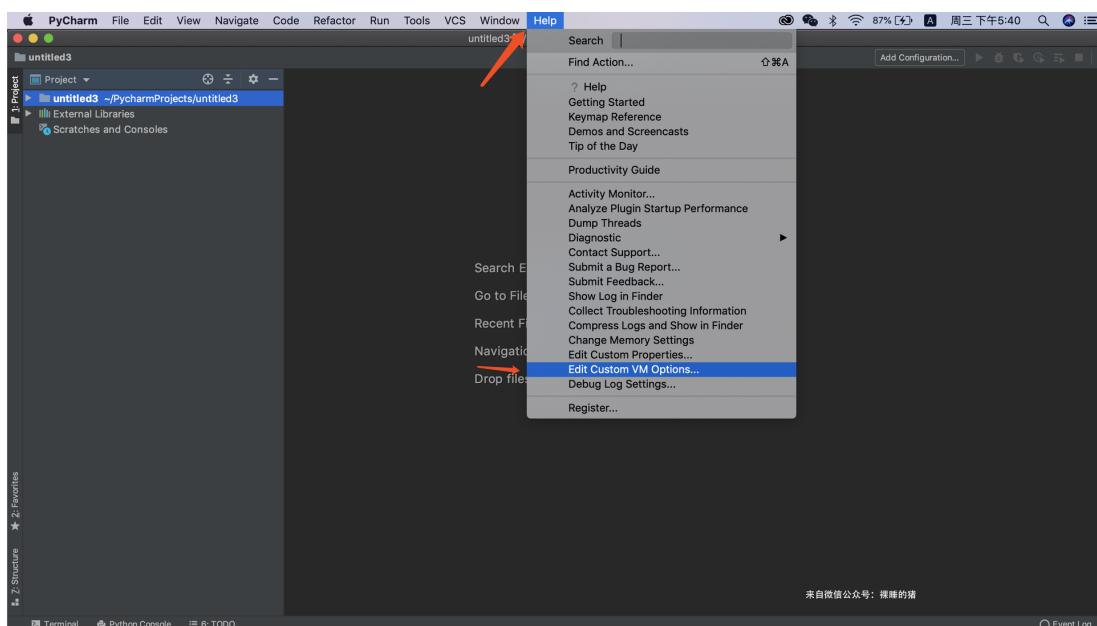
我们现创建一个空项目



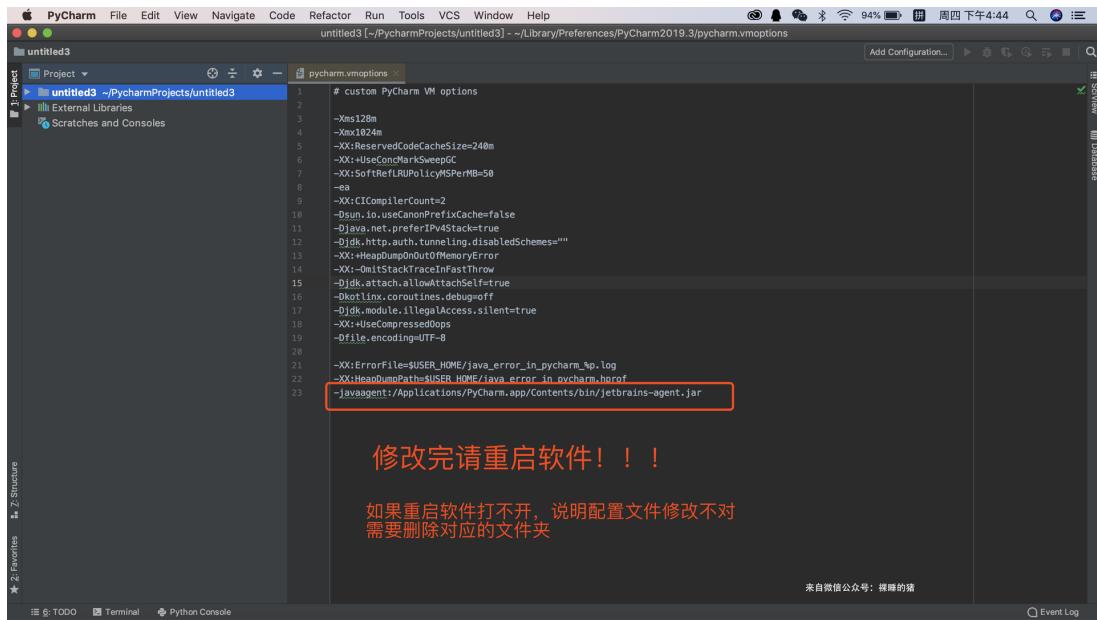
如果你的Create按钮是灰色，那点击图上部那个向右的小三角形，选择你安装Python就可以了！



进入到项目界面后，点击Pycharm最上面的菜单栏中的“Help”->“Edit Custom VM Options ...”。



在打开的vmoptions编辑窗口末行添加：-javaagent:你pycharm的安装目录\jetbrains-agent.jar请仔细检查补丁路径是否正确，如果错误则会出现pycharm打不开的情况。



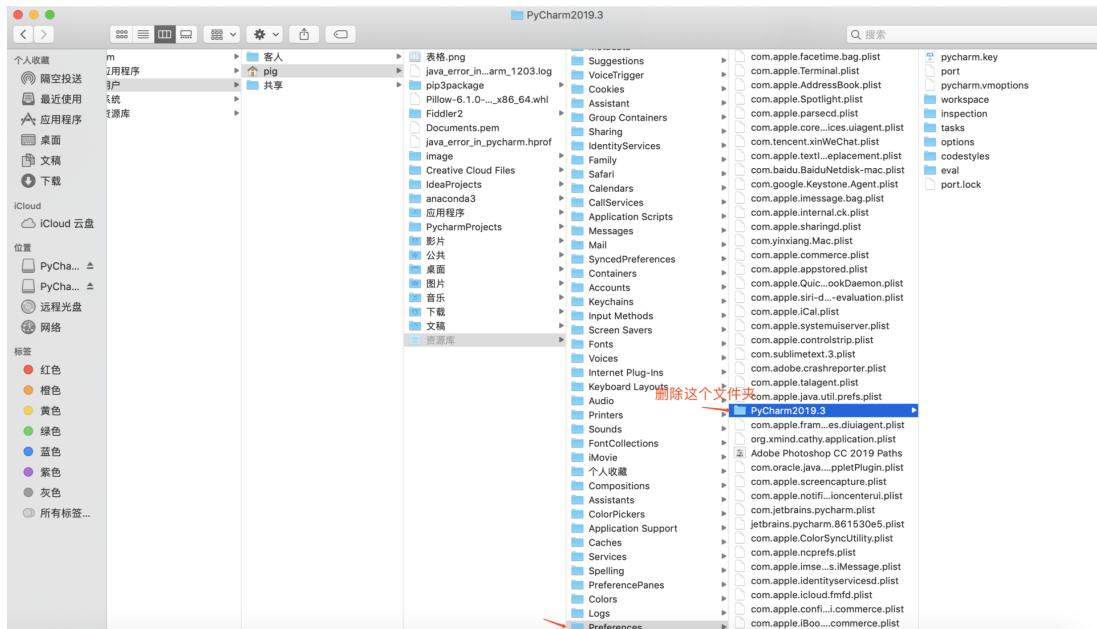
修改完配置文件之后切记重启PyCharm软件

如果修改完打不开软件，或者提示没有jdk等问题，这时候可以删除用户目录下的pycharm文件夹，注意这个文件夹是隐藏目录！

windows: C:\Users\用户名\

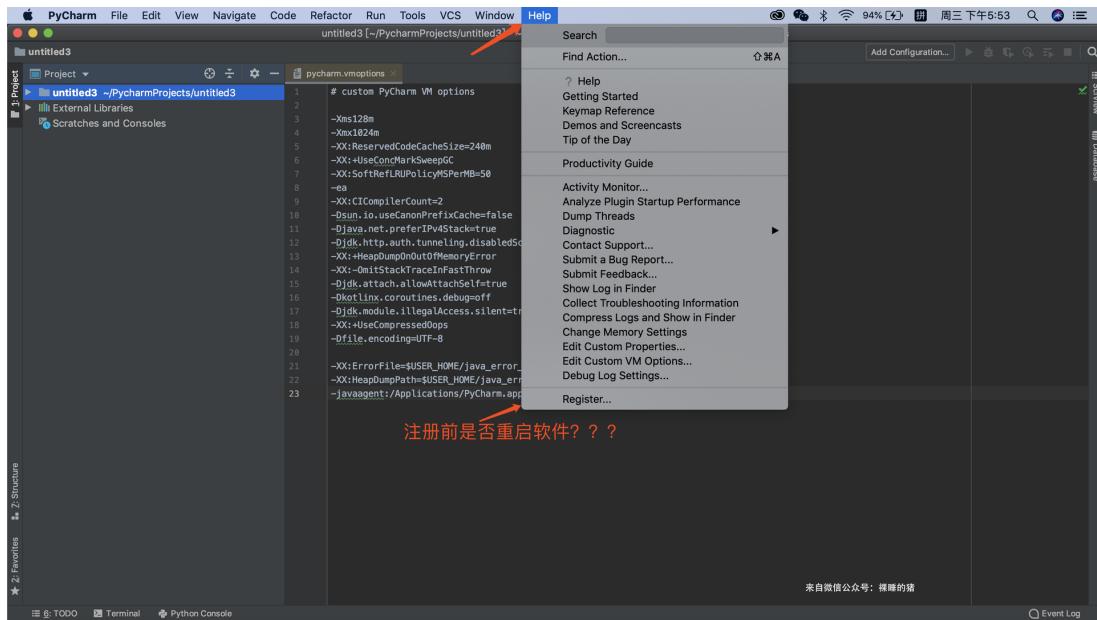
macos: ~/Library/Preferences/

ubuntu: ~/.



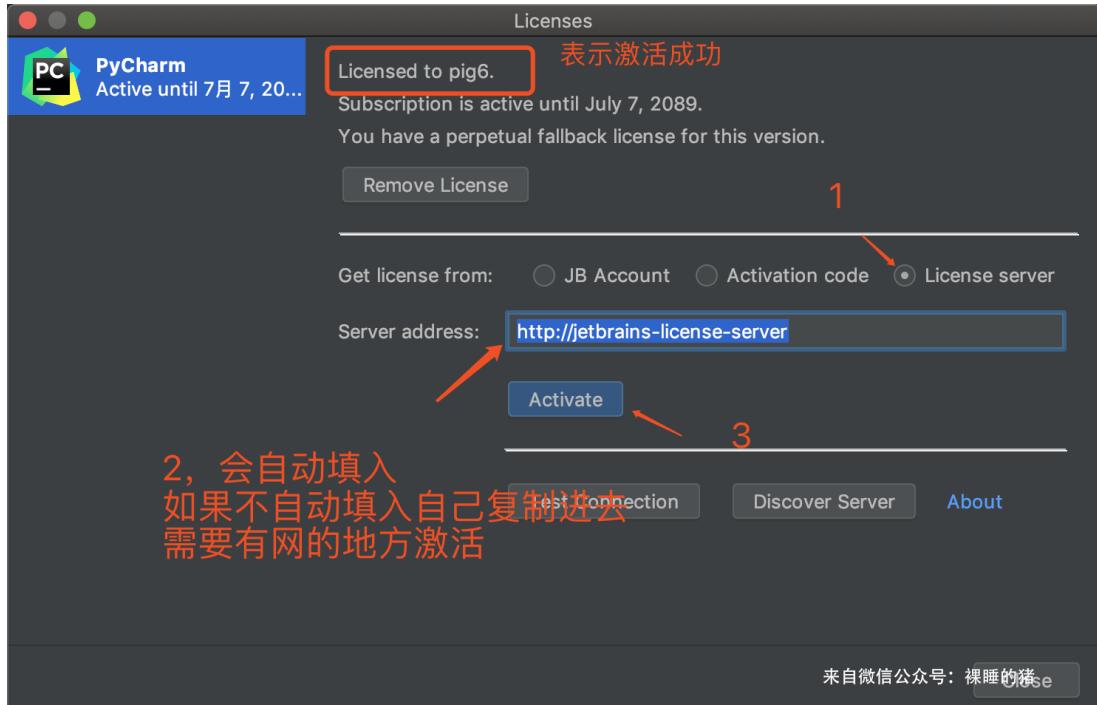
4. 输入激活码

修改完配置文件之后重启pycharm，点击菜单栏中的“Help”->“Register ...”



选择最后一种License server激活方式，地址填入：<http://jetbrains-license-server>（应该会自动填上），或者点击按钮：“Discover Server”来自动填充地址，完成激活。

注意：服务器激活需要联网！

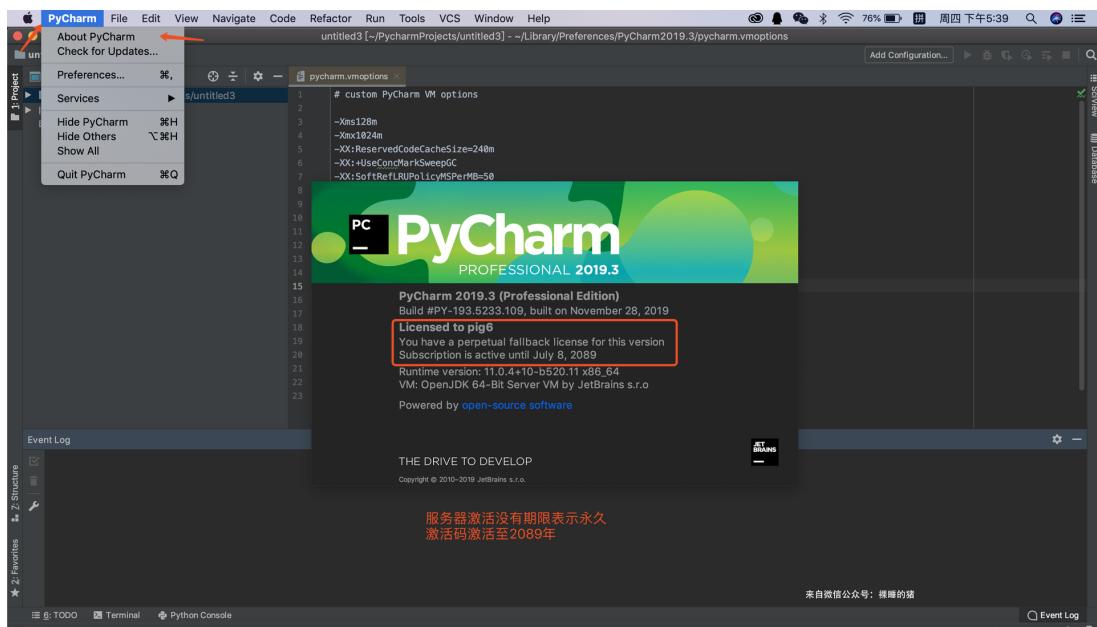


5.查看有效期

当你激活完毕后，PyCharm右下角会有个Registration小长条提示框，大致的内容为： You copy is Licensed to XXX意思就会告诉你：兄弟，你已经激活成功了，激活码的许可来源是：XXX。

查看有效期的步骤为点击： Help->About这里可以看到你的pycharm的版本号、许可来源、有效期、以及一些环境

服务器激活是没有期限的，即为永久有效。



9. 简单试一下

```
1 print("hello,python!")
```

输出

```
1 hello,python!
```

一个简单的例子，猜大小

```
1 print("*" * 30)
2 print("欢迎进入骰子游戏")
3 print("*" * 30)
4
5 username = input('请输入用户名\n')
6 money = 0
7
```

```
8 answer = input('确定进入游戏吗(y/n)\n')
9
10 import random
11
12 if answer == "y":
13     while money < 2:
14         n = int(input("金币不足, 请充值, 必须是100的倍数, 如200, 500"))
15         if n % 100 == 0 and n > 0:
16             money = (n // 100) * 30
17
18     print("当前剩余游戏币是: {}, 玩一局扣除两个游戏币".format(money))
19
20     print("进入游戏")
21     while True:
22         t1 = random.randint(1, 6)
23
24         t2 = random.randint(1, 6)
25
26         money -= 2
27         guess = input("请输入大小:")
28         if ((t1 + t2) > 6 and guess == "大") or ((t1 + t2) <= 6 and guess == "小"):
29             print("结果为:{}{}".format(t1+t2))
30             print("恭喜答对了")
31
32     else:
33         print("结果为:{}{}".format())
34         print("输了")
35     answer = input("是否再来一局游戏, 要扣除2枚游戏币? (y/n)")
36
37     if answer!="y" or money<2:
38         print("退出游戏啦")
39         break
40
41
```

当然语法还是要熟悉一点的，这中间我就看了一点，觉得还是直接边写边熟悉。

10. 安装工具包。

Python 2.7.9+ 或 Python 3.4+ 以上版本都自带 pip 工具。

pip 官网：<https://pypi.org/project/pip/>

你可以通过以下命令来判断是否已安装：

查看pip版本

```
1 pip --version
```

1遇到的问题是 pip 也许是对应的python2.7

 pip3 对应的是python 3.xx。

如果我们要同时开发多个应用程序，那这些应用程序都会共用一个Python，就是安装在系统的Python 3。

每个应用可能需要各自拥有一套“独立”的Python运行环境。virtualenv就是用来为一个应用创建一套“隔离”的Python运行环境。

2 安装virtualenv

```
1 pip3 install virtualenv
2 pip install virtualenvwrapper
```

2。查看版本

```
1 virtualenv --version
```

3. 新建一个项目，需要一套独立的Python运行环境。

```
1 第一步    创建目录
2 →      mkdir test
3
4 →      cd test
5
6 第二步    创建一个独立的Python运行环境，命名为venv
```

```
7
8 → virtualenv --no-site-packages venv
9
10 Using base prefix '/usr/local/.../Python.framework/Versions/3
11 New python executable in venv/bin/python3.4
12 Also creating executable in venv/bin/python
13 Installing setuptools, pip, wheel...done.
14
15 参数--no-site-packages 意思是：
16 已经安装到系统Python环境中的所有第三方包都不会复制过来，
17 这样，我们就得到了一个不带任何第三方包的“干净”的Python运行环境。
18
19 第三步 新的venv这个Python环境创建好了，进入创建好的环境。
20
21 进入
22
23 → source /venv/bin/activate
24 (venv) →
25 前面会带一个 (venv) 前缀。
26
27 退出环境
28
29 → deactivate
30
31
32
33 第四步 安装需要的包。
34
35 pip install -i https://pypi.doubanio.com/simple/ Django
36
37 如果出错 使用下面命令
38
39 pip install -i https://pypi.doubanio.com/simple/ --trusted-
40
```

```
41 参数解释:
42
43 -i https://pypi.doubanio.com/simple/      # -i指定使用源 (-i ==
44 --trusted-host pypi.doubanio.com          # --trusted-host 表示添加
45
46 你还可以使用其他源
47 阿里云 http://mirrors.aliyun.com/pypi/simple/
48 中国科技大学 https://pypi.mirrors.ustc.edu.cn/simple/
49 豆瓣(douban) http://pypi.doubanio.com/simple/
50 清华大学 https://pypi.tuna.tsinghua.edu.cn/simple/
51 中国科学技术大学 http://pypi.mirrors.ustc.edu.cn/simple/
52
```

还有就是 python源可以修改 成 你想要的豆瓣源 阿里云源~~~

百度搜索 。

如果 退出， 就回到了正常环境。

小结

virtualenv为应用提供了隔离的Python运行环境，解决了不同应用间多版本的冲突问题。

11. Django 学习。

1.安装 django 框架。

个人理解，对错不知道！！！

可以在 虚拟环境外面安装，也可以在 新创建的虚拟环境里面安装。

注意： 电脑系统中 如果有 2 个python版本 python2.7 和 python 3.8

 你用的python2.7， 应该是pip。

 你用的是python3 应该是pip3。

还要注意的是 你可以把python2.7卸载，也可以更改一下 配置环境 启动的时候就是python3.

这样用的时候 就不会混淆了。

指定版本安装 -- 1.11.11。

```
1 pip install django==1.11.11
```

升级到 指定版本 (2.2)

```
1 pip install --upgrade django==2.2
```

如果安装慢，或者安装失败 用豆瓣源安装django。

```
1 pip install -i https://pypi.doubanio.com/simple/ Django
```

如果出错使用下面命令

```
1 pip install -i https://pypi.doubanio.com/simple/ --trusted-host pypi.doubanio.com
```

参数解释

```
1 -i https://pypi.doubanio.com/simple/      # -i指定使用源 (-i ==  
2 --trusted-host pypi.doubanio.com          # --trusted-host 表示添加信任
```

其他源

```
1 -----  
2 阿里云 http://mirrors.aliyun.com/pypi/simple/  
3 中国科技大学 https://pypi.mirrors.ustc.edu.cn/simple/  
4 豆瓣(douban) http://pypi.doubanio.com/simple/  
5 清华大学 https://pypi.tuna.tsinghua.edu.cn/simple/  
6 中国科学技术大学 http://pypi.mirrors.ustc.edu.cn/simple/  
7 -----
```

如何设置成默认源

```
1 http://pip.readthedocs.org/en/latest/user_guide.html#config-file
```

安装完成~~~

查看Django 是否安装成功

1.在创建的虚拟环境查看

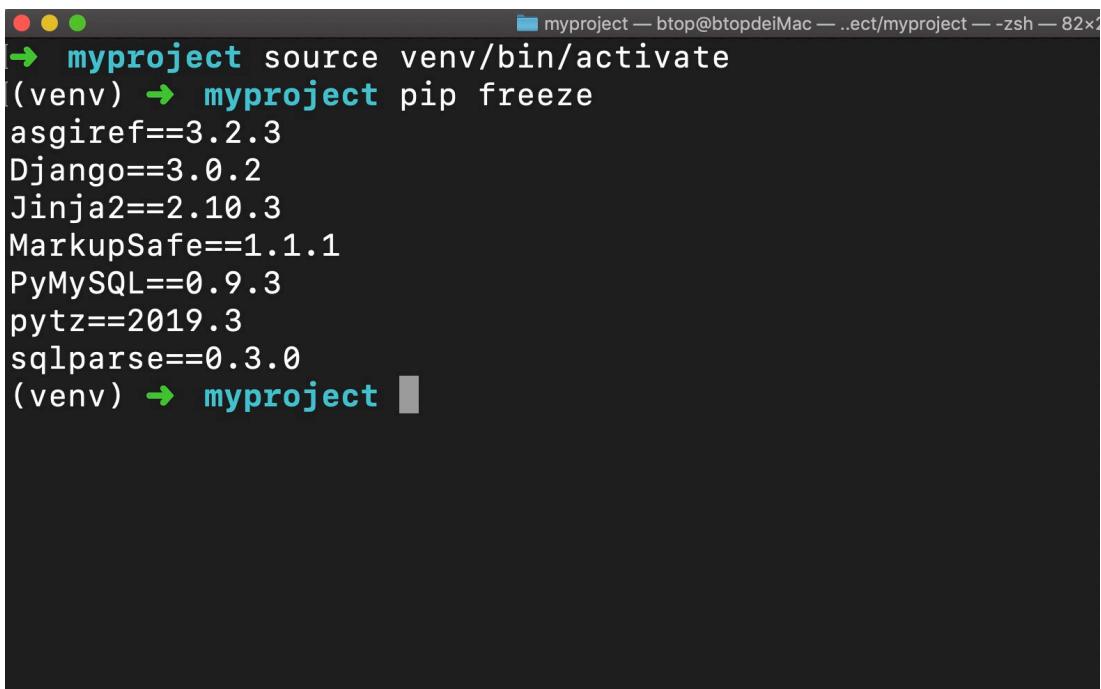
```
1 进入到 创建的 虚拟环境中  
2 输入 下面两个命令  
3 source venv/bin/activate  
4 pip freeze
```

5

1 还可以使用 这个命令查看 版本。

2 pip show django

3



```
myproject - btop@bttopdeiMac - ..ect/myproject - -zsh - 82x24
→ myproject source venv/bin/activate
(venv) → myproject pip freeze
asgiref==3.2.3
Django==3.0.2
Jinja2==2.10.3
MarkupSafe==1.1.1
PyMySQL==0.9.3
pytz==2019.3
sqlparse==0.3.0
(venv) → myproject
```

退出 虚拟环境的 命令是 deactivate

安装Django 的版本是 3.23

因为项目已经创建虚拟环境了，所以没必要在去看正常环境有没有安装django了。如果在正常环境下开发项目，还是需要安装Django。

2.这是 没在虚拟环境，正常环境下。

1 打开终端

```
btop@bttopdeiMac: ~ - zsh - 82x25
→ ~ python ←
Python 2.7.16 (default, Jun 19 2019, 07:40:37)
[GCC 4.2.1 Compatible Apple LLVM 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import django ← 这里输入的python, 显示的是python2.7版本
>>> django.get_version() 导入 django 显示出版本1.11.11
'1.11.11' ← 因为两个版本共存, 应该是我用pip 下载的时候
>>> exit() 对应的是python2.7. (不清楚, 暂时这样理解, 哈哈)
→ ~ python3
Python 3.8.1 (v3.8.1:1b293b6006, Dec 18 2019, 14:08:53)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import django ←
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'django'
>>> exit()
→ ~ [ ] ← 这里输入的python3, 却导包不成功, 应该是没下载把, 或许是不能这样操作。我个人觉得应该是用 pip3 install django==2.2 下载django
```

1

12. Django常用命令

进入到新创建的虚拟环境中。

1.新建django 项目

```
1 django-admin.py startproject project_name (project_name是项目名)
```

当前目录下会多出一个文件夹, 进入到创建的项目下。输入下面命令, 启动项目。

2.启动 django 项目

```
1 python manage.py runserver #默认使用8000端口
```

命令后面还可以指定参数:

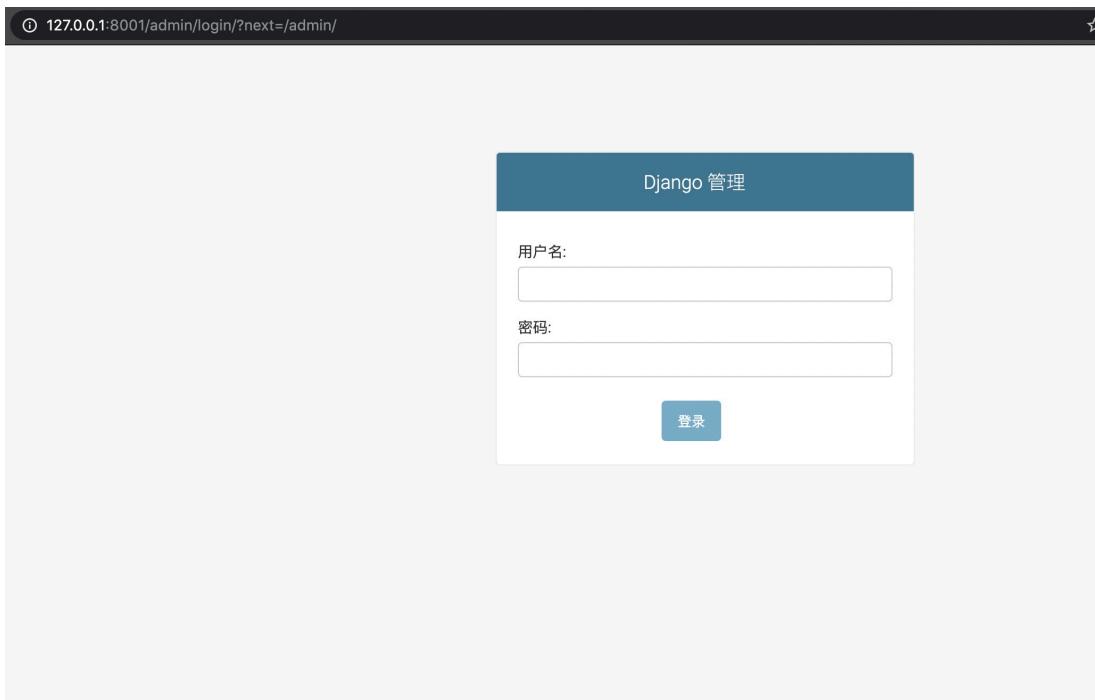
```
1 python manage.py runserver 8888 #8888为新指定的端口
2 python manage.py runserver 127.0.0.1:8000 #还可以指定IP和端口,
```

```
1 (venv) ➔ mysite python manage.py runserver 8080
2 Watching for file changes with StatReloader
3 Performing system checks...
4
```

```
5 System check identified no issues (0 silenced).
6 January 15, 2020 - 20:09:19
7 Django version 3.0.2, using settings 'mysite.settings'
8 Starting development server at http://127.0.0.1:8080/
9 Quit the server with CONTROL-C.
```

打开浏览器，输入 <http://127.0.0.1:8080/> 就可以访问了 不过会提示报错

访问这个 `http://127.0.0.1:8080/admin` 这个接口，就会出现



ok了~~~~

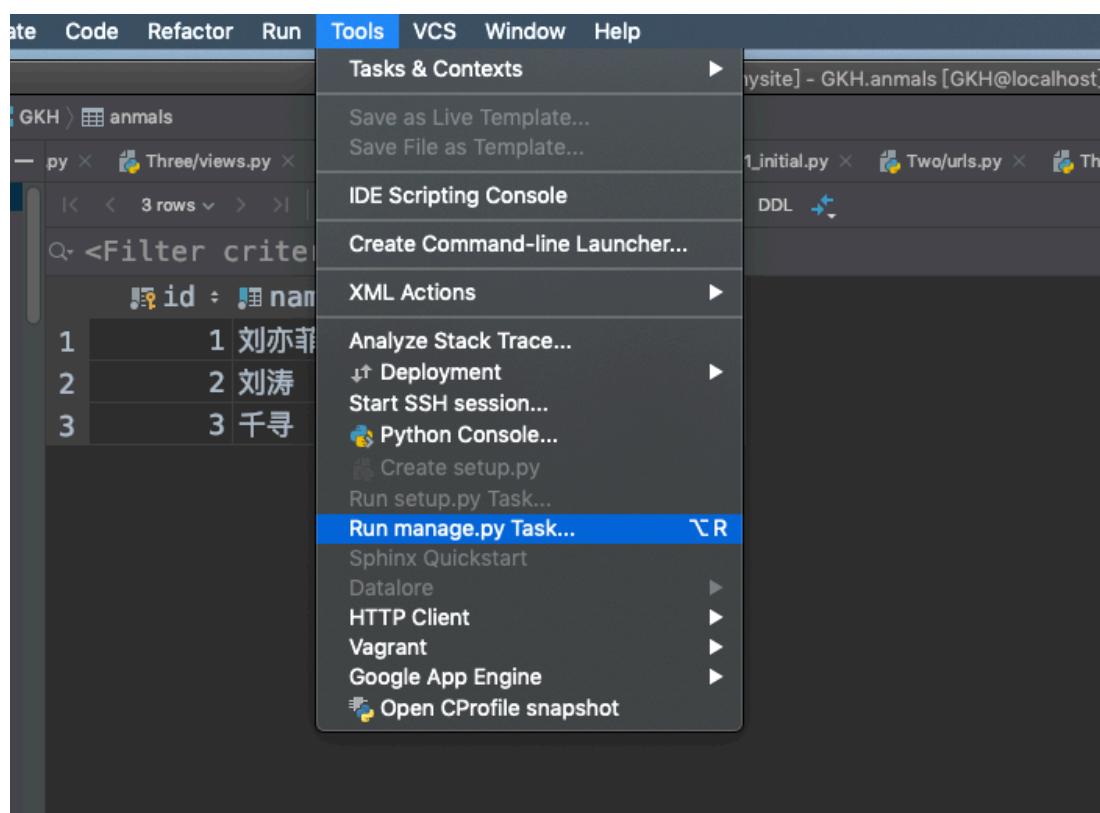
3. 创建App

一个Django项目可以分为很多个APP，用来隔离不同功能模块的代码。

1. 命令行创建

```
1 python manage.py startapp app      (app是名称，可以换)
```

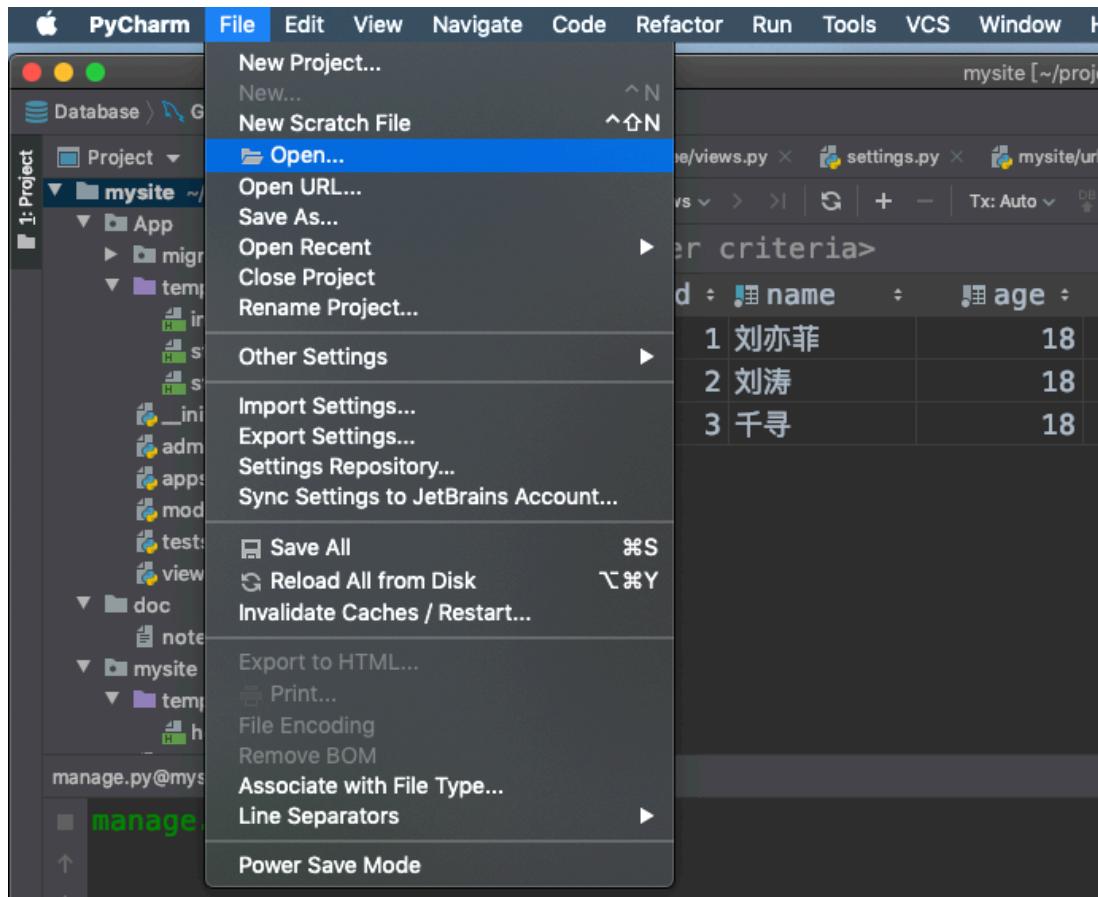
2.pycharm 创建（不过要先打开pycharm，选择一下虚拟环境，可以先看下一步）



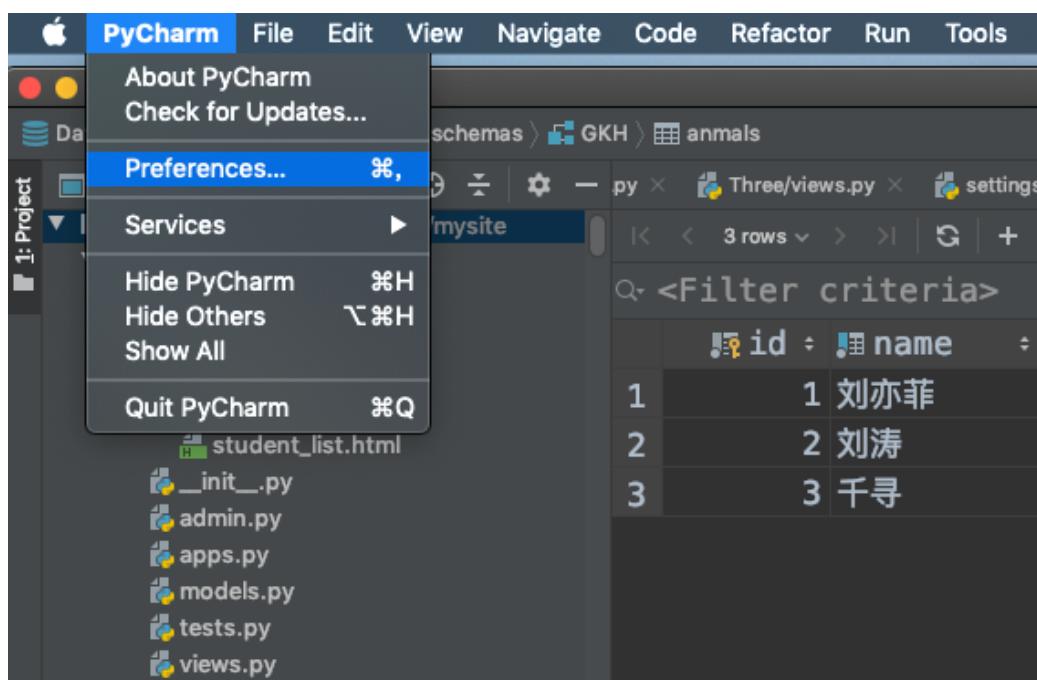
```
1 startapp app (app 可以 换 其他 名字)
```

打开pycharm 编辑器，选择虚拟环境。

先打开创建的django项目。

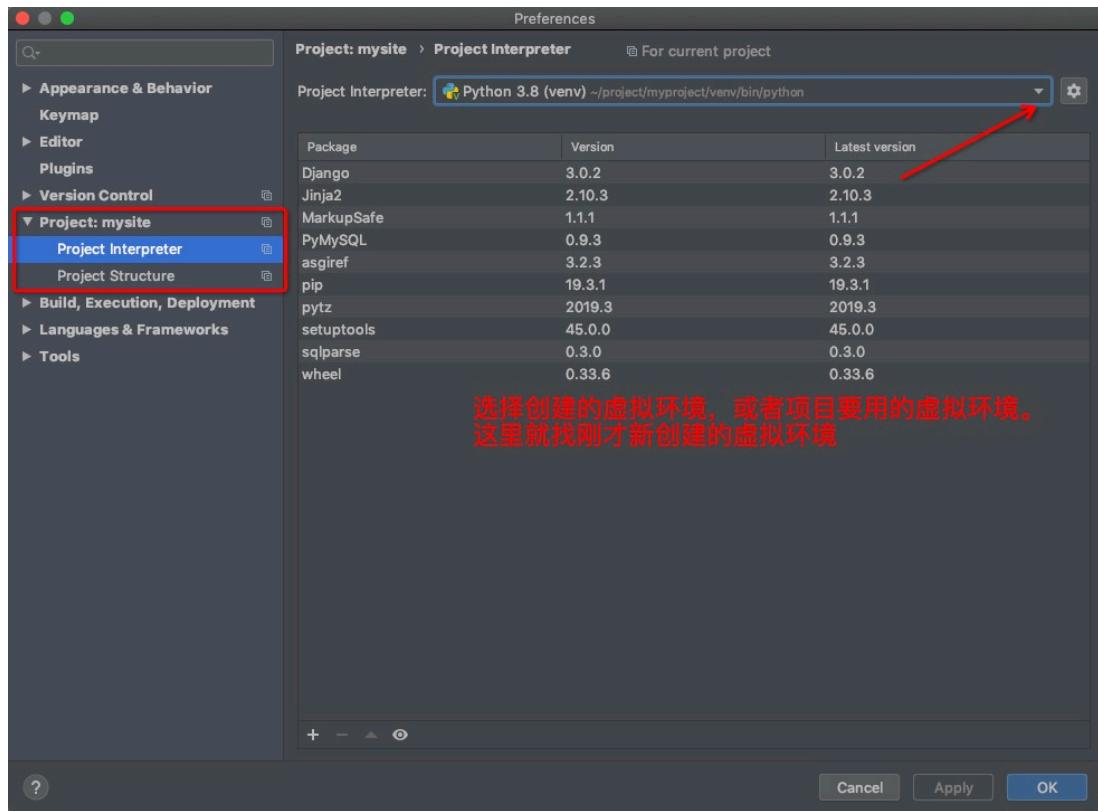


选择设置

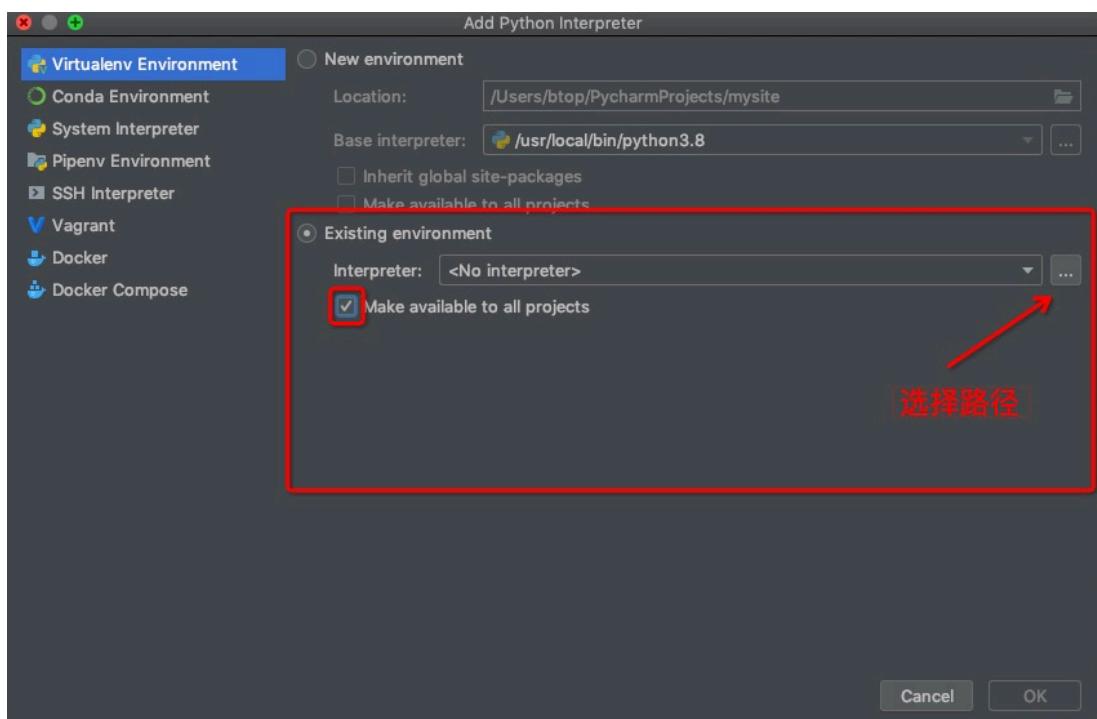


mac 是 preferences, windows 是 setting. (应该是)

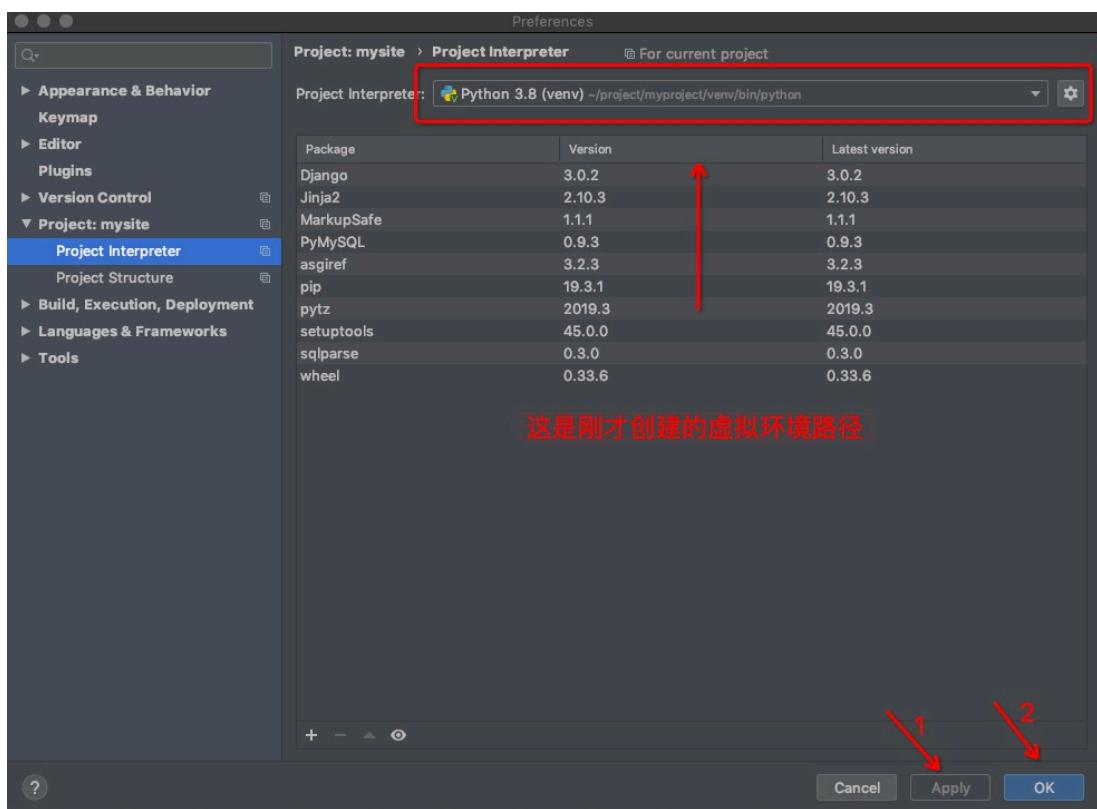
如果没有 就点右上角的那个齿轮。



点齿轮按钮，选择 Add.



这是我新建项目的那个虚拟环境路径。



现在就可以 在pycharm 里 运行项目了，虽然没写，但是启动项目，可以访问刚才的那个/admin 接口。

```
Terminal: Local × Local (2) × Local (3) × +
(venv) ➔ mysite python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 15, 2020 - 20:45:31
Django version 3.0.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

Error: That port is already in use.
(venv) ➔ mysite
```

出现这种情况，是端口被占用了。

windows

- 1 打开终端 或者Cmd 输入
- 2 netstat -ano | findstr "8080"

```
C:\Users\Administrator>netstat -ano | findstr 8080
TCP    0.0.0.0:8080          0.0.0.0:0              LISTENING      7704
TCP    [::]:8080            [::]:0                LISTENING      7704
C:\Users\Administrator>
```

杀死进程

- 1 tasklist | findstr 7704

mac

1.查找占用端口

- 1 lsof i:8000

```
1 (venv) ➔ mysite lsof -i:8000
2 COMMAND      PID USER      FD      TYPE             DEVICE SIZE/OFF NO
3 python3.8  7416 btop      5u    IPv4  0x7c353a68fea4eb73        0t0  T
4
```

杀死进程

```
1 kill -9 7416 或者 kill 7416
```

重新运行

```
1 (venv) ➔ mysite python manage.py runserver
2 Watching for file changes with StatReloader
3 Performing system checks...
4
5 System check identified no issues (0 silenced).
6 January 15, 2020 - 20:58:24
7 Django version 3.0.2, using settings 'mysite.settings'
8 Starting development server at http://127.0.0.1:8000/
9 Quit the server with CONTROL-C.
10
```

--

123

234

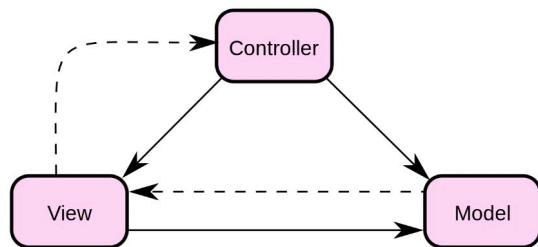
13.开始写接口。

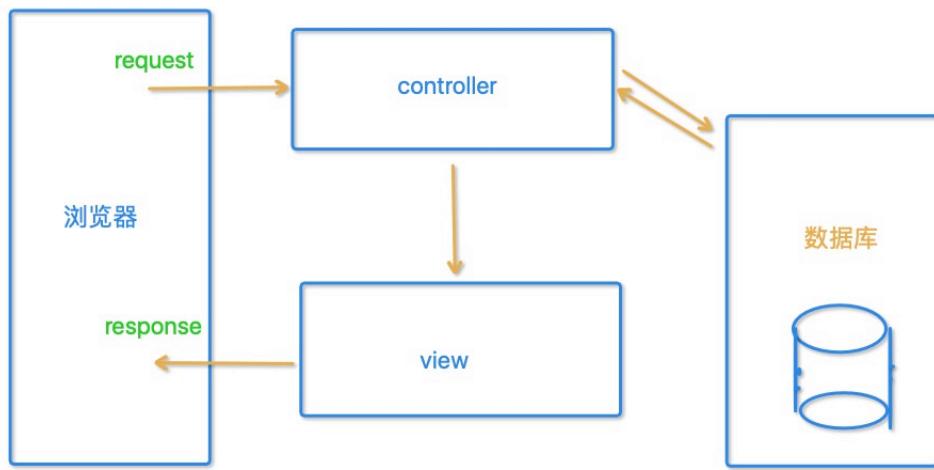
要理解 MVC 模式

MVC 模式

MVC 模式代表 Model-View-Controller（模型-视图-控制器）模式。这种模式用于应用程序的分层开发。

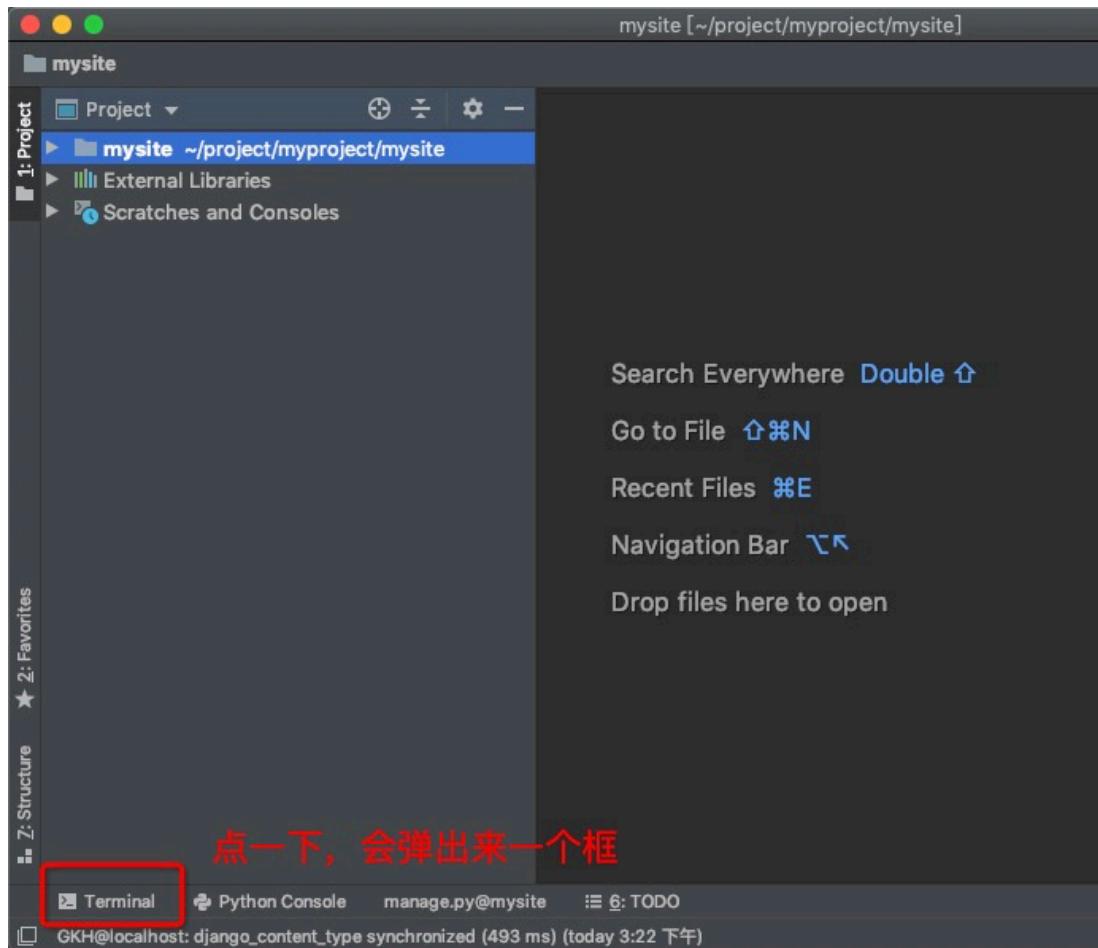
- **Model**（模型） - 模型代表一个存取数据的对象或 JAVA POJO。它也可以带有逻辑，在数据变化时更新控制器。
- **View**（视图） - 视图代表模型包含的数据的可视化。
- **Controller**（控制器） - 控制器作用于模型和视图上。它控制数据流向模型对象，并在数据变化时更新视图。它使视图与模型分离开。





应该是这样，我随便画的。

1.新建一个app文件夹（在pycharm中用 Terminal）



123

2.输入

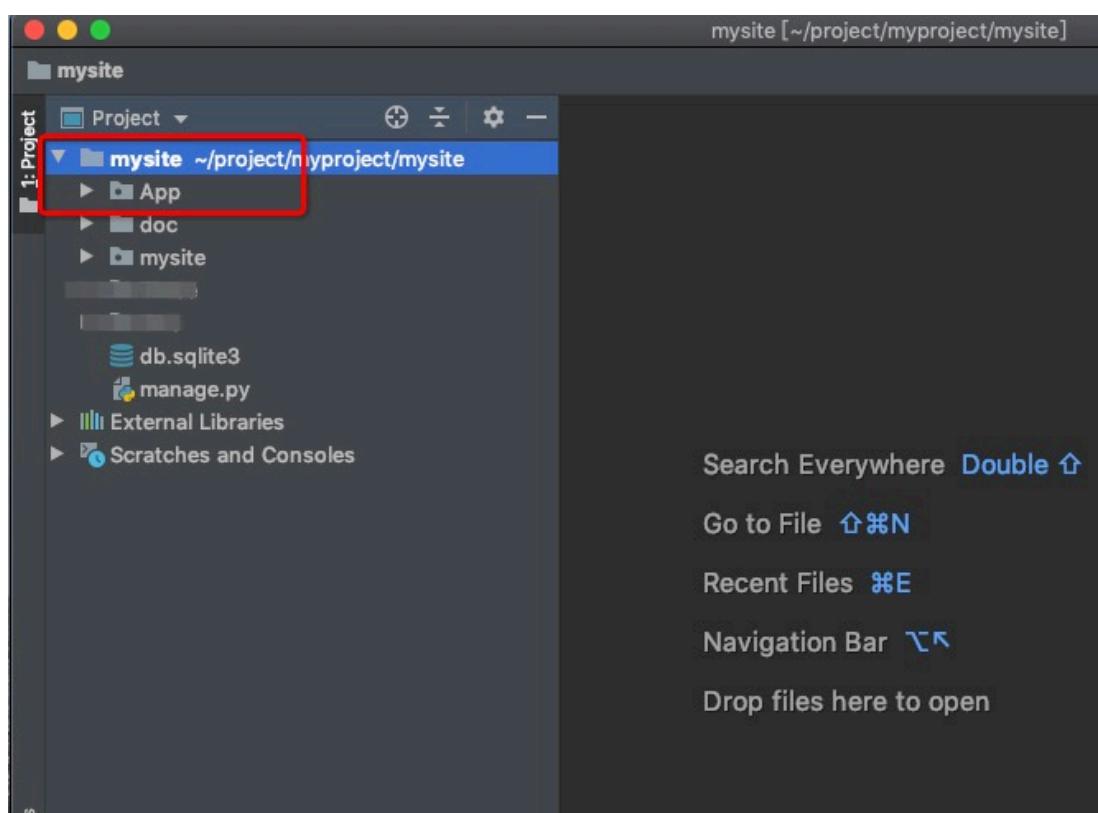
```
1 python manage.py startapp app_name      (app_name 可以换成其他名字)  
2  
3 我新建一个App  
4 python manage.py startapp App
```

注意：如果失败，可能是目录不对。在虚拟环境创建Django项目的那一层目录。

比如你的Django 项目名称是 mysite，那就在这层目录下。

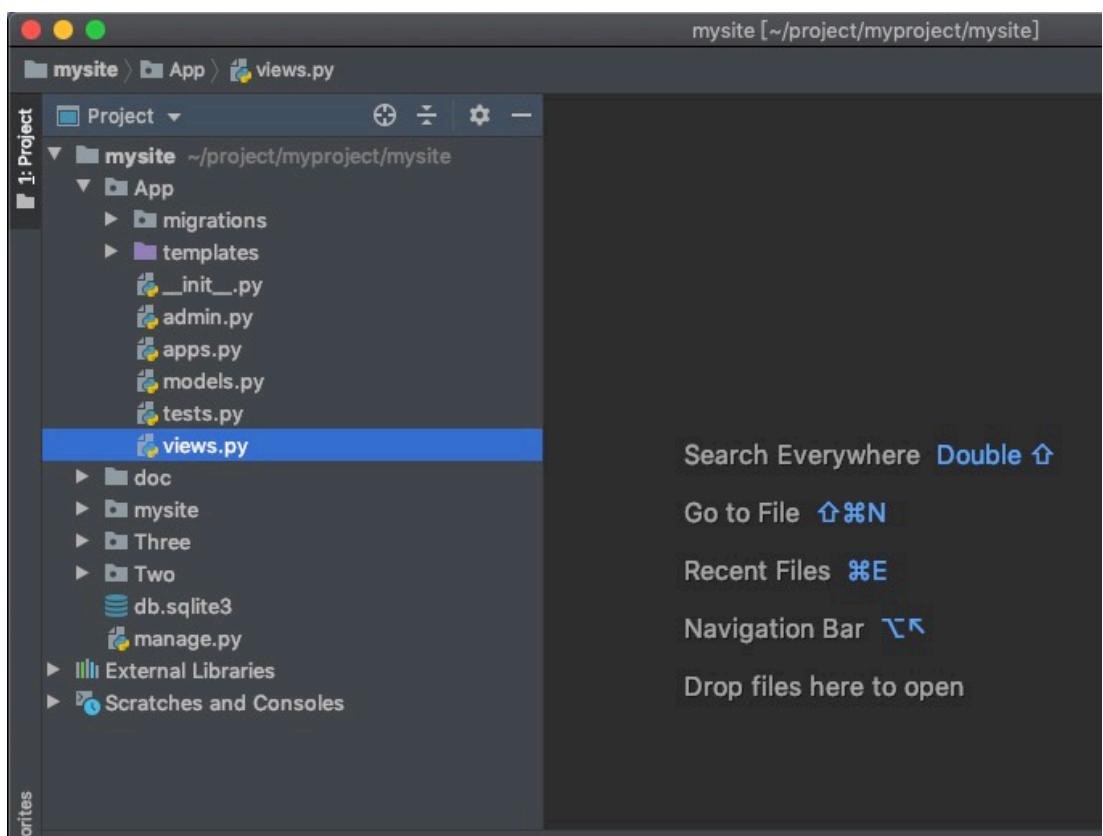
```
1 (venv) ➔ mysite pwd (windows是 dir)  
2 /Users/btop/project/myproject/mysite  
3  
4 (venv) ➔ mysite python manage.py startapp app_name
```

3.生成一个App文件夹

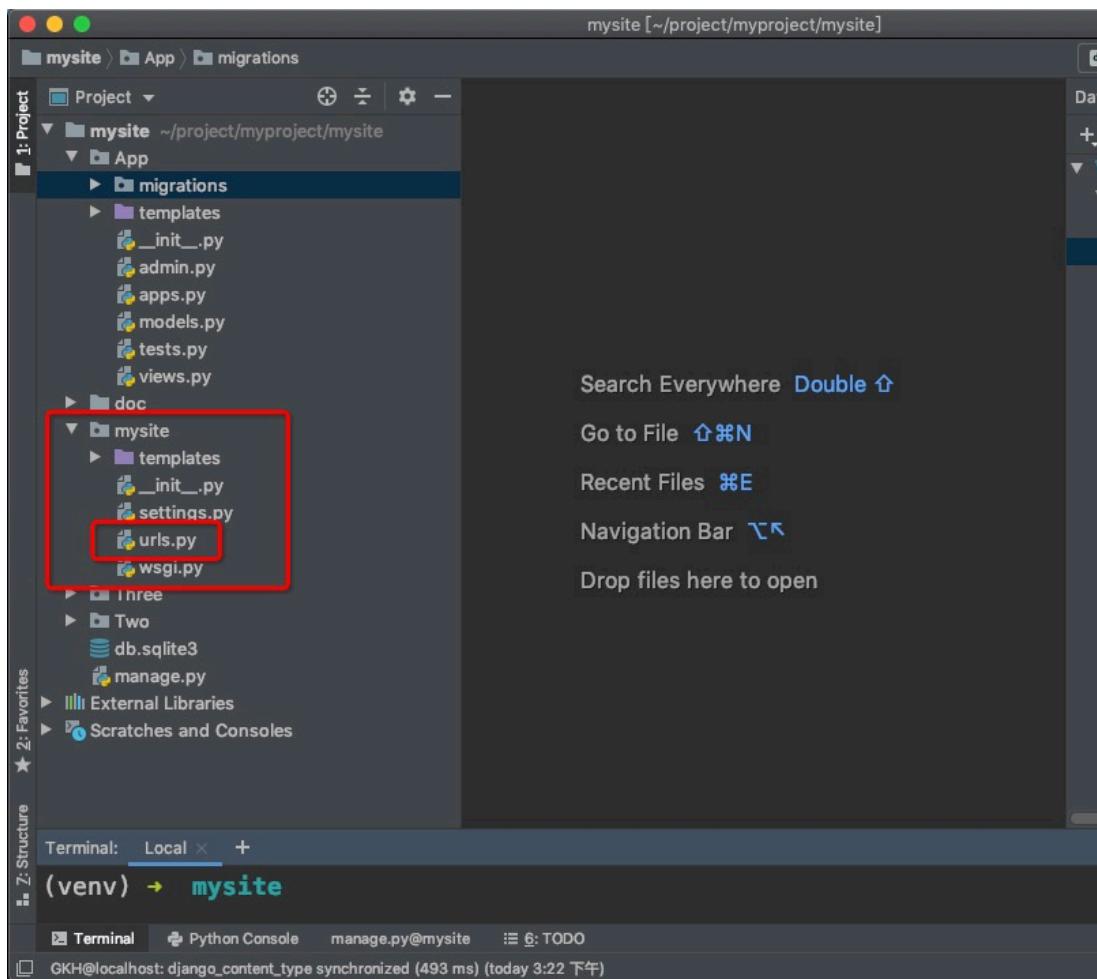


4.打开App 会弹出很多文件

1。templates文件夹 是我新建的文件夹，不是生成的。



2. 打开 urls文件夹



3. 里面是写url 请求的文件

```

2. Add a URL to urlpatterns: url(r'^blog/', include('blog.urls'))
"""
import ...

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^hello/', views.hello), 新建一个/hello 接口

```

项目运行，请求/admin接口

新建一个/hello 接口

123

4. 上面新建的一个 /hello 接口 里面参数有 view.hello

导包

The screenshot shows a file structure for a Django project named 'mysite'. Inside the 'mysite' directory, there is an 'App' folder containing several files: migrations, templates, __init__.py, admin.py, apps.py, models.py, tests.py, and views.py. The 'views.py' file is highlighted with a red arrow pointing from the file tree. In the main code editor area, the 'urls.py' file is open. It contains the following code:

```
1. Import the include() function from django.urls
2. Add a URL to urlpatterns: url(r'^blog/', include
   ...
from django.conf.urls import url, include
from django.contrib import admin

from App import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^hello/$', views.hello), hello是函数名称
]
```

Annotations in red highlight specific parts of the code:

- A box around the line "from App import views" is labeled "App是包名".
- A box around the line "url(r'^hello/\$', views.hello)" is labeled "views.hello是函数名称".
- A box around the word "views" is labeled "views是文件名".

5.开始写函数

The screenshot shows the 'views.py' file being edited. The file structure on the left is identical to the previous screenshot. The code in 'views.py' is as follows:

```
from django.http import HttpResponseRedirect
from django.shortcuts import render

# Create your views here.

def hello(request):
    return HttpResponseRedirect("世界，你好")
```

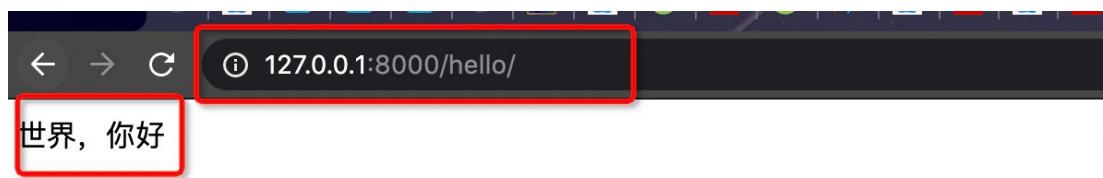
Annotations in red highlight specific parts of the code:

- A box around the imports "from django.http import HttpResponseRedirect" and "from django.shortcuts import render" is labeled "from django.http import HttpResponseRedirect" and "from django.shortcuts import render".
- A box around the function definition "def hello(request):" is labeled "def hello(request):".

6.运行

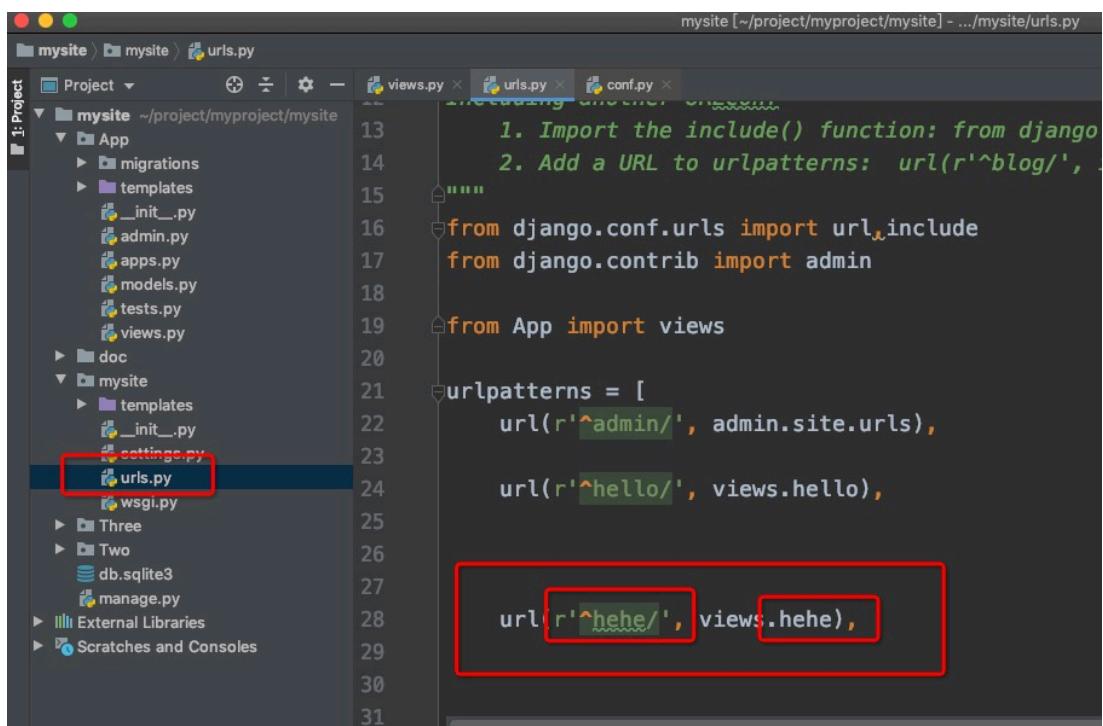
```
1 python manage.py runserver
```

打开浏览器，输入请求url.



7在写一个

1. 打开urls 文件



```
mysite [~/project/myproject/mysite] - .../mysite/urls.py
views.py x urls.py x conf.py x
Project views.py urls.py conf.py
mysite ~/project/myproject/mysite
  App
    migrations
    templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      views.py
  doc
  msite
    templates
      __init__.py
      settings.py
      urls.py
      wsgi.py
  Three
  Two
    db.sqlite3
    manage.py
External Libraries
Scratches and Consoles
```

```
13     1. Import the include() function: from django
14     2. Add a URL to urlpatterns: url(r'^blog/',
15     """)
16     from django.conf.urls import url, include
17     from django.contrib import admin
18
19     from App import views
20
21     urlpatterns = [
22         url(r'^admin/', admin.site.urls),
23
24         url(r'^hello/$', views.hello),
25
26         url(r'^hehe/$', views.hehe),
27
28     ]
29
30
31
```

代码。

```
1 urlpatterns = [
2     url(r'^admin/', admin.site.urls),
3
4     url(r'^hello/$', views.hello),
5
6     url(r'^hehe/$', views.hehe),
7 ]
```

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window, which lists the project structure. The 'App' folder under 'mysite' is expanded, showing files like migrations, templates, __init__.py, admin.py, apps.py, models.py, tests.py, and views.py. The 'views.py' file is selected and shown in the main editor area. The code in 'views.py' is:

```
from django.http import HttpResponseRedirect
from django.shortcuts import render

# Create your views here.

def hello(request):
    return HttpResponseRedirect("世界, 你好")

def hehe(request):
    return render(request, 'home.html')
```

A red box highlights the 'from django.shortcuts import render' import statement and the 'hehe' view definition. A red arrow points from the 'hehe' view definition to the 'home.html' template name in the 'render' call.

3.在 App 文件夹下 创建 templates 文件夹 放html文件。

为什么是紫色的。因为紫色好看。哈哈

新建templates文件夹 是白色的。需要设置一下。

The screenshot shows the PyCharm IDE interface. The project structure remains the same, but now there is a new 'templates' folder inside the 'App' folder. This folder is highlighted with a red box. The 'views.py' code is identical to the previous screenshot:

```
from django.http import HttpResponseRedirect
from django.shortcuts import render

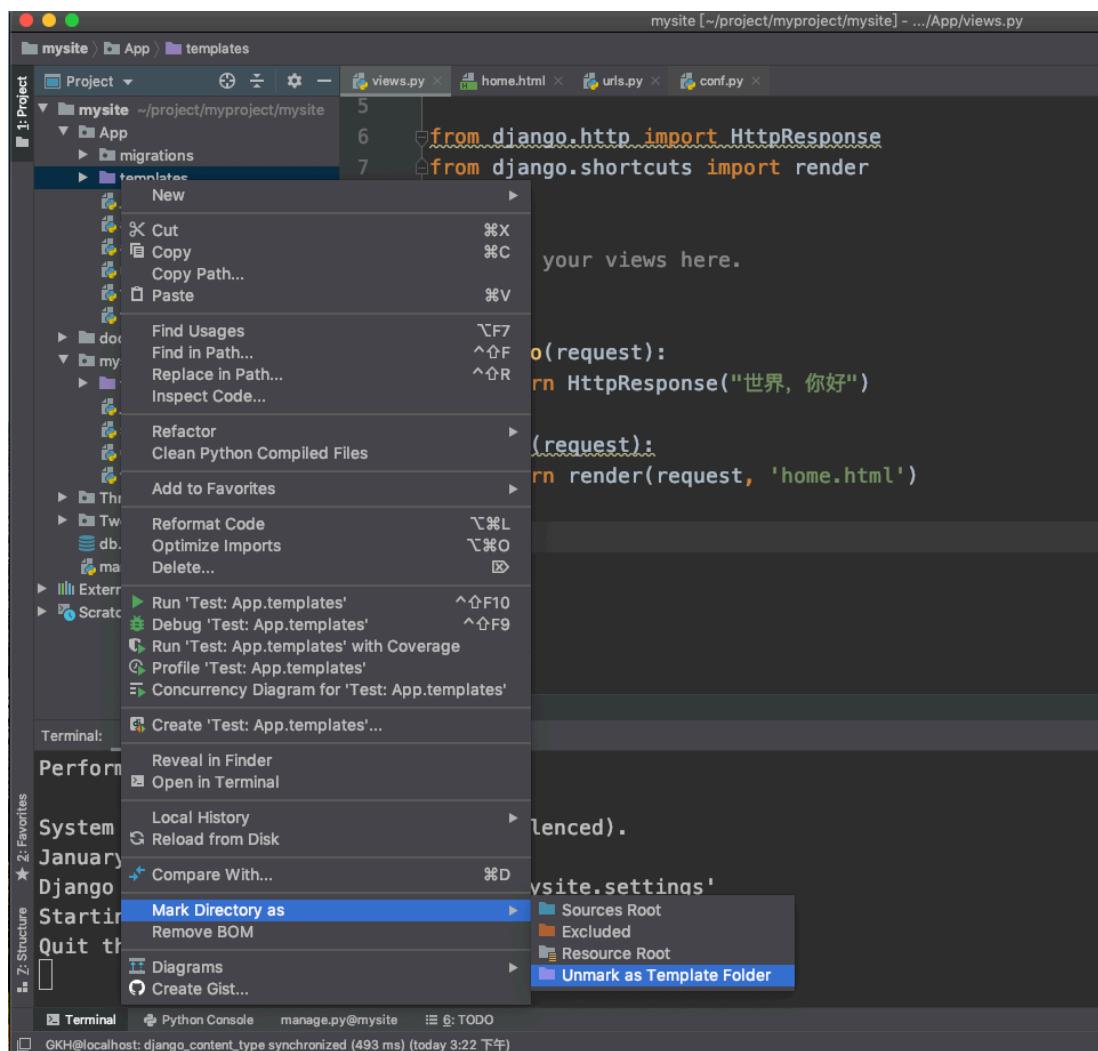
# Create your views here.

def hello(request):
    return HttpResponseRedirect("世界, 你好")

def hehe(request):
    return render(request, 'home.html')
```

4.添加

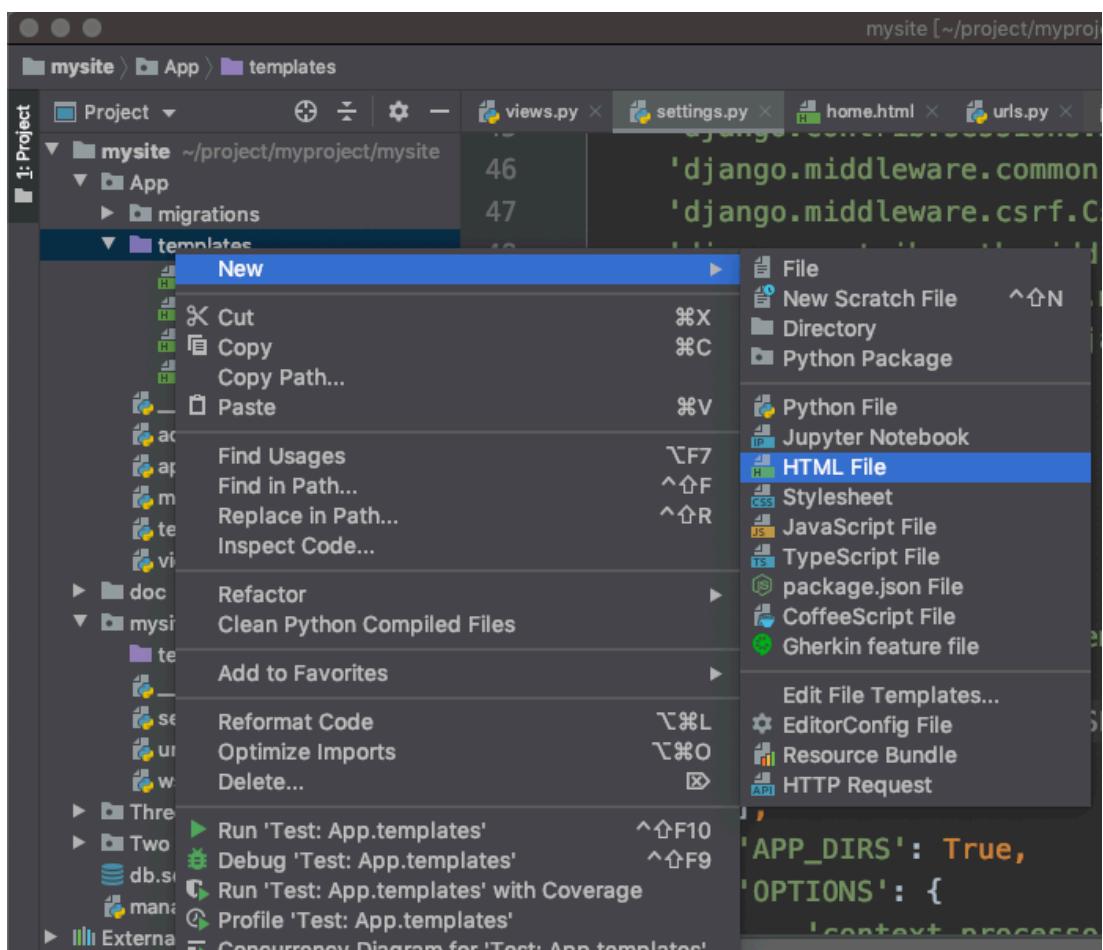
在templates文件夹上右击一下。



这样就变成紫色了。

5.写 html 文件。

在 templates文件夹下 新建一个 home.html 文件。



home.html文件

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure under 'mysite'. The main window shows the code editor for 'home.html' with the following content:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>home</title>
    </head>
    <body>
        <ul>
            <li>世界, 真美好! </li>
            <li>男未婚</li>
            <li>女未嫁</li>
            <li>勾搭勾搭</li>
            <li>多融洽</li>
        </ul>
    </body>
</html>
```

代码

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>home</title>
6  </head>
7  <body>
8  <ul>
9      <li>世界, 真美好! </li>
10     <li>男未婚</li>
11     <li>女未嫁</li>
12     <li>勾搭勾搭</li>
13     <li>多融洽</li>
14
15 </ul>
16 </body>
17 </html>
```

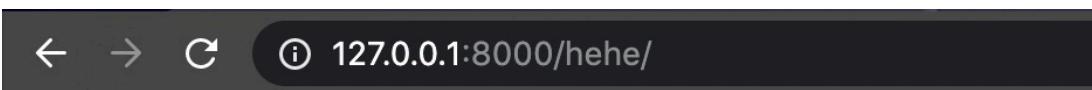
快捷键

输入 `ul>li*5` 然后 按 tab 键

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>home</title>
6   </head>
7   <body>
8     <ul>
9       <li>世界, 真美好! </li>
10      <li>男未婚</li>
11      <li>女未嫁</li>
12      <li>勾搭勾搭</li>
13      <li>多融洽</li>
14    </ul>
15
16    ul>li*4 ← 快速生成
17  </body> *后面 跟数字 *5 *9
18 </html>
```

6.运行 访问

```
1 在浏览器 输入 127.0.0.1:8000/hehe
```



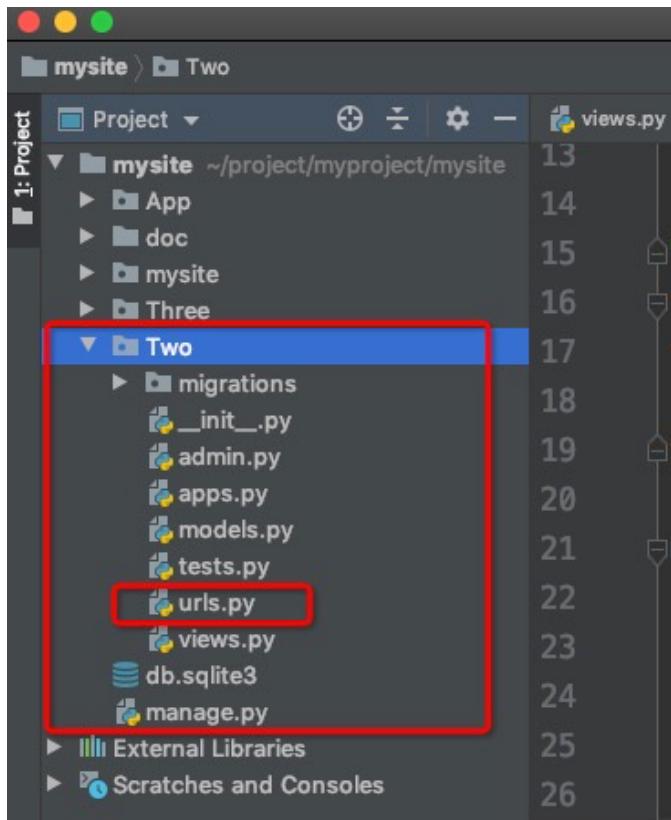
- 世界, 真美好!
- 男未婚
- 女未嫁
- 勾搭勾搭
- 多融洽

14.路由

1.新建一个App，就会在文件夹下生成一个 Two文件夹。

```
1 python manage.py startapp Two
```

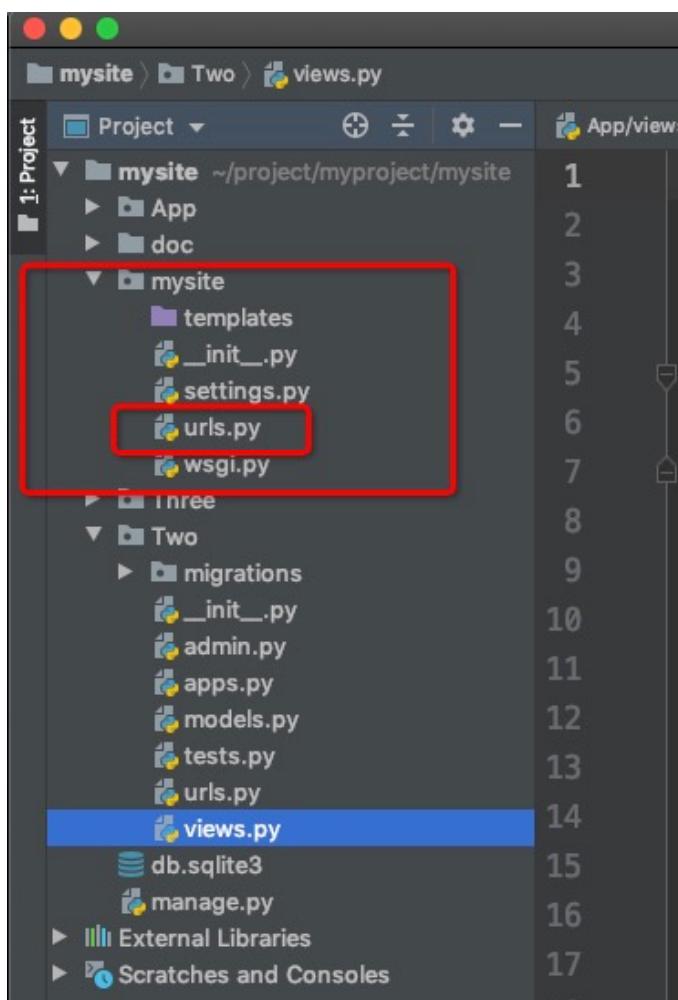
2.不过没有 urls文件，需要自己新建。



3.urls文件的代码

```
1 from django.conf.urls import url
2
3 from Two import views
4
5 urlpatterns = [
6     url(r'^home/$', views.index),
7
8
9 ]
```

4.打开 创建django的 文件夹里 的 urls 文件



5. 把url(r'^two/\$', include('Two.urls')) , 加进去。

后面是新建的Two 下的 urls 文件。

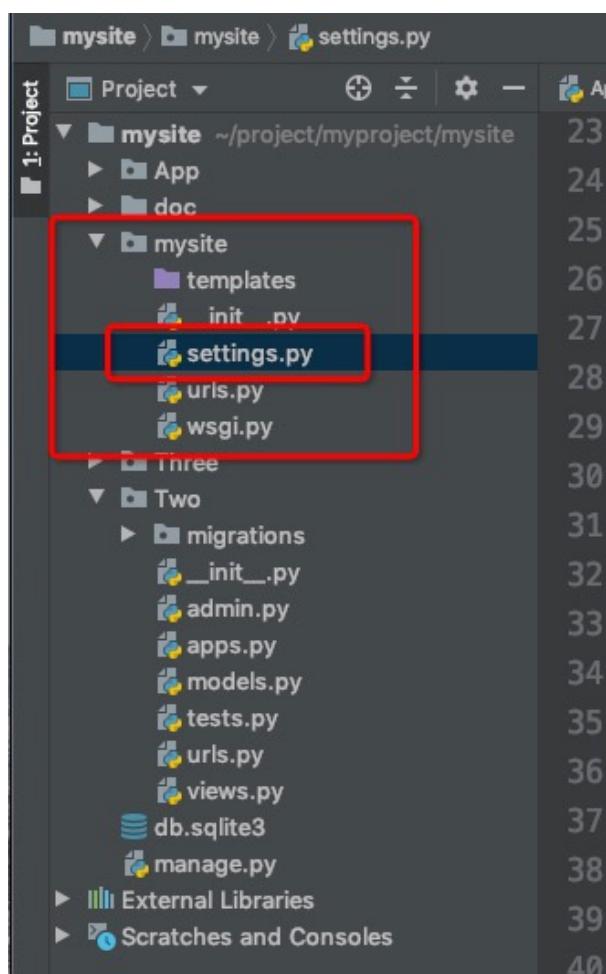
```
1 urlpatterns = [
2     url(r'^admin/', admin.site.urls),
3     url(r'^hello/$', views.hello),
4     url(r'^hehe/$', views.hehe),
5     url(r'^two/$', include('Two.urls')), 也就是这一行。
6 ]
7
8
9     这里的      '^two/' ,      可以改成其他的
```

6. 打开 Two 下的 views 文件 (这里的 view 文件是刚才新创建 App 下的 views 文件)

```
1
2
3 from django.http import HttpResponse
4 # Create your views here.
5 def index(request):
6     pass
7     return HttpResponse("200")
8
```

7. 一切都写好了，要把 Two 加到 settings.py 里面去。

打开 settings.py



The screenshot shows the PyCharm IDE interface. On the left is the project tree, which includes a 'mysite' project containing 'App', 'Two', and 'Three' applications. Each application has its own 'models.py', 'views.py', and 'urls.py' files. The 'mysite' directory also contains 'templates', 'init_.py', and 'settings.py'. The right side of the screen shows the content of 'settings.py'. Two specific sections are highlighted with red boxes:

```
25 DEBUG = True  
26 # 代表通配 所有人都可以访问  
27 ALLOWED_HOSTS = ["*"]  
28  
29 # Application definition  
30  
31 INSTALLED_APPS = [  
32     'django.contrib.admin',  
33     'django.contrib.auth',  
34     'django.contrib.contenttypes',  
35     'django.contrib.sessions',  
36     'django.contrib.messages',  
37     'django.contrib.staticfiles',  
38     'App',  
39     'Two',  
40     'Three'  
41 ]
```

8。运行。

```
1 python manage.py runserver
```

访问浏览器，也可以用postman工具。



200 要加上/two， 在urls写的是/two

14.操作数据库。

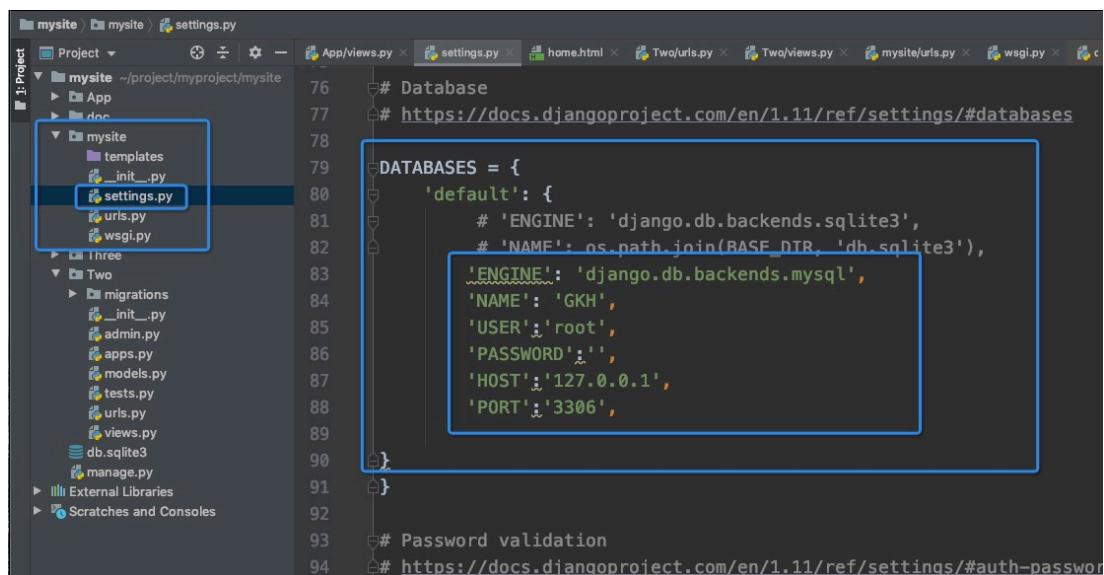
1. 安装数据库驱动。我用的是 PyMySQL

```
1 pip install pymysql -i https://pypi.douban.com/simple
```

查看是否安装成功

```
1 pip freeze      输入 命令
2
3 (venv) → mysite pip freeze
4 asgiref==3.2.3
5 Django==3.0.2      #这个是Django
6 Jinja2==2.10.3
7 MarkupSafe==1.1.1
8 PyMySQL==0.9.3      #这个就是操作数据库的驱动
9 pytz==2019.3
10 sqlparse==0.3.0
11 (venv) → mysite
12
13
```

2. 打开文件 django 项目文件下的 settings.py 文件。



The screenshot shows the PyCharm IDE interface with the project 'mysite' open. The 'Project' tool window on the left displays the directory structure:

- mysite (selected)
- App
- doc
- mysite (containing __init__.py, templates, urls.py, wsgi.py)
- Two (containing migrations, __init__.py, admin.py, apps.py, models.py, tests.py, urls.py, views.py, db.sqlite3, manage.py)
- Three
- External Libraries
- Scratches and Consoles

The 'Code' tool window on the right shows the 'settings.py' file with the following code:

```
# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        # 'ENGINE': 'django.db.backends.sqlite3',
        # 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'GKH',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}

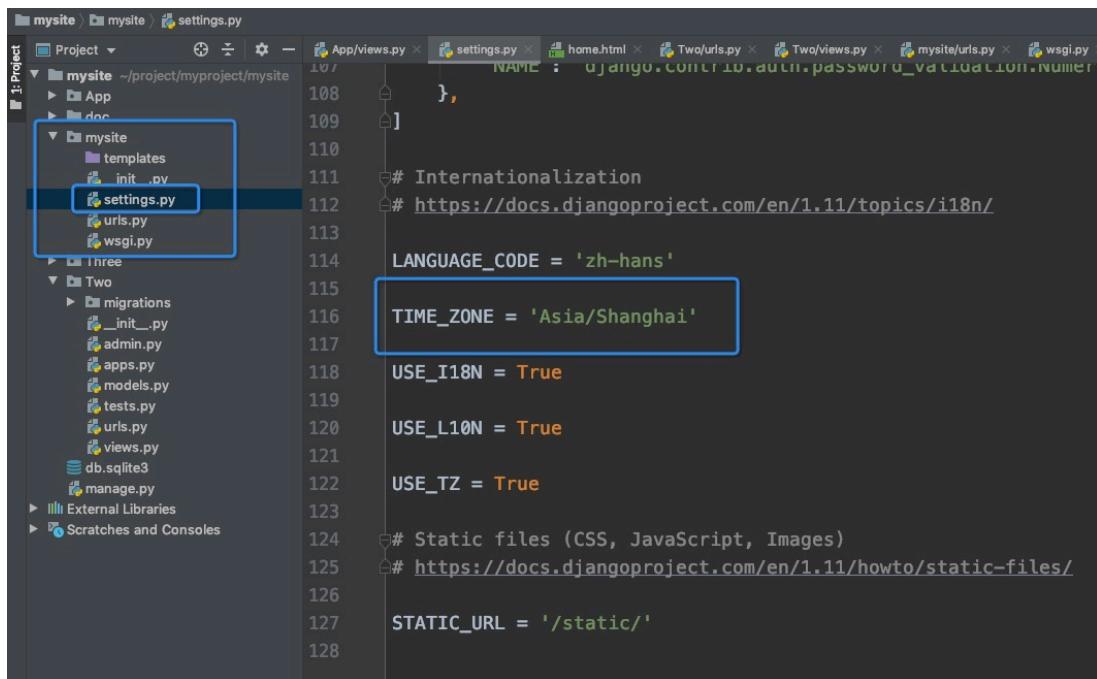
# Password validation
# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-validators
```

A blue rectangular box highlights the MySQL database configuration section.

代码

```
1 DATABASES = {  
2     'default': {  
3         # 'ENGINE': 'django.db.backends.sqlite3',  
4         # 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
5         'ENGINE': 'django.db.backends.mysql',  
6         'NAME': 'GKH',           #数据库名称  
7         'USER': 'root',        #账号  
8         'PASSWORD': '',       #密码  
9         'HOST': '127.0.0.1',   #ip  
10        'PORT': '3306',       #端口  
11    }  
12 }
```

改下时间



```
1 Project : mysite / mysite / settings.py  
2 mysite ~/project/myproject/mysite  
3 108     },  
4 109     ]  
5 110     # Internationalization  
6 111     # https://docs.djangoproject.com/en/1.11/topics/i18n/  
7 112     LANGUAGE_CODE = 'zh-hans'  
8 113     TIME_ZONE = 'Asia/Shanghai' // This line is highlighted by a blue box.  
9 114     USE_I18N = True  
10    USE_L10N = True  
11    USE_TZ = True  
12    # Static files (CSS, JavaScript, Images)  
13    # https://docs.djangoproject.com/en/1.11/howto/static-files/  
14    STATIC_URL = '/static/'  
15  
16 124  
17 125  
18 126  
19 127  
20 128
```

不改时间 运行的时候 这里 显示的时候 会有8小时误差。

```
System check identified no issues (0 silenced).
January 15, 2020 - 21:41:40
Django version 3.0.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
/Users/btop/project/myproject/mysite/App/views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
```

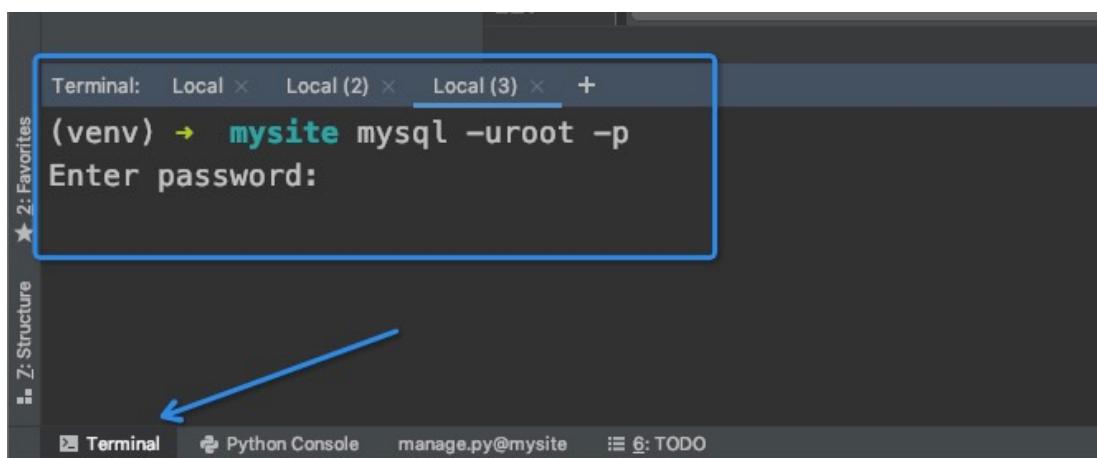
打开项目下 `_init_.py` 文件 加入下面代码。

```
1 import pymysql
2 pymysql.install_as_MySQLdb()
```

3.先去mysql数据库 新建一个库,也可以用已经存在库。 数据库的名字就是 刚才写在 `settings.py` 文件里的名字。

```
1 DATABASES = {
2     'default': {
3         # 'ENGINE': 'django.db.backends.sqlite3',
4         # 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
5         'ENGINE': 'django.db.backends.mysql',
6         'NAME': 'GKH',           #数据库名称      这个就是 数据库 名字
7         'USER': 'root',          #账号          root
8         'PASSWORD': '',          #密码
9         'HOST': '127.0.0.1',    #ip
10        'PORT': '3306',          #端口
11
12 }
```

4.打开mysql 数据库



Terminal: Local × Local (2) × Local (3) × +

(venv) → mysite mysql -uroot -p

Enter password:

2: Favorites

Z: Structure

Terminal Python Console manage.py@mysite 6: TODO

```
1 mysql -uroot -p
2 Enter password:
```

查看所有数据库信息

```
1 mysql> show databases; //show databases;
```

新建一个数据库

```
1 mysql> create database Winter; #Winter 换成 GHK 或者其他Name。
2 Query OK, 1 row affected (0.10 sec)
3
```

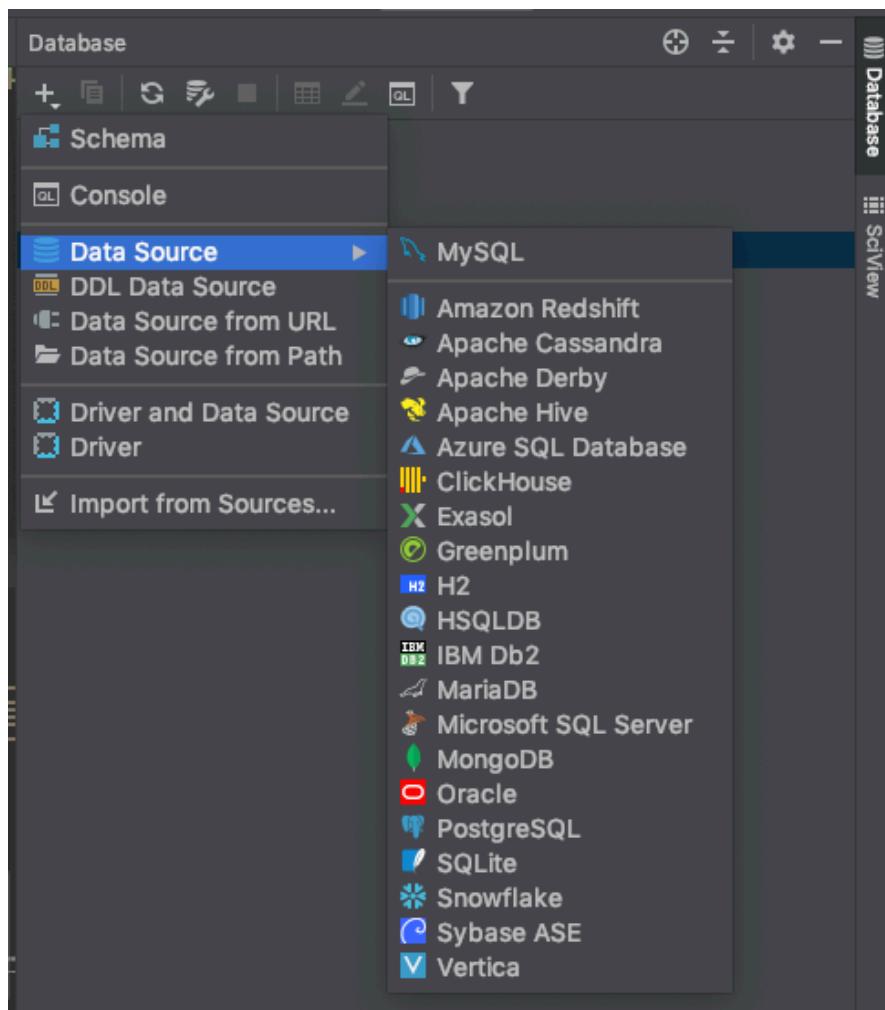
1在是打发

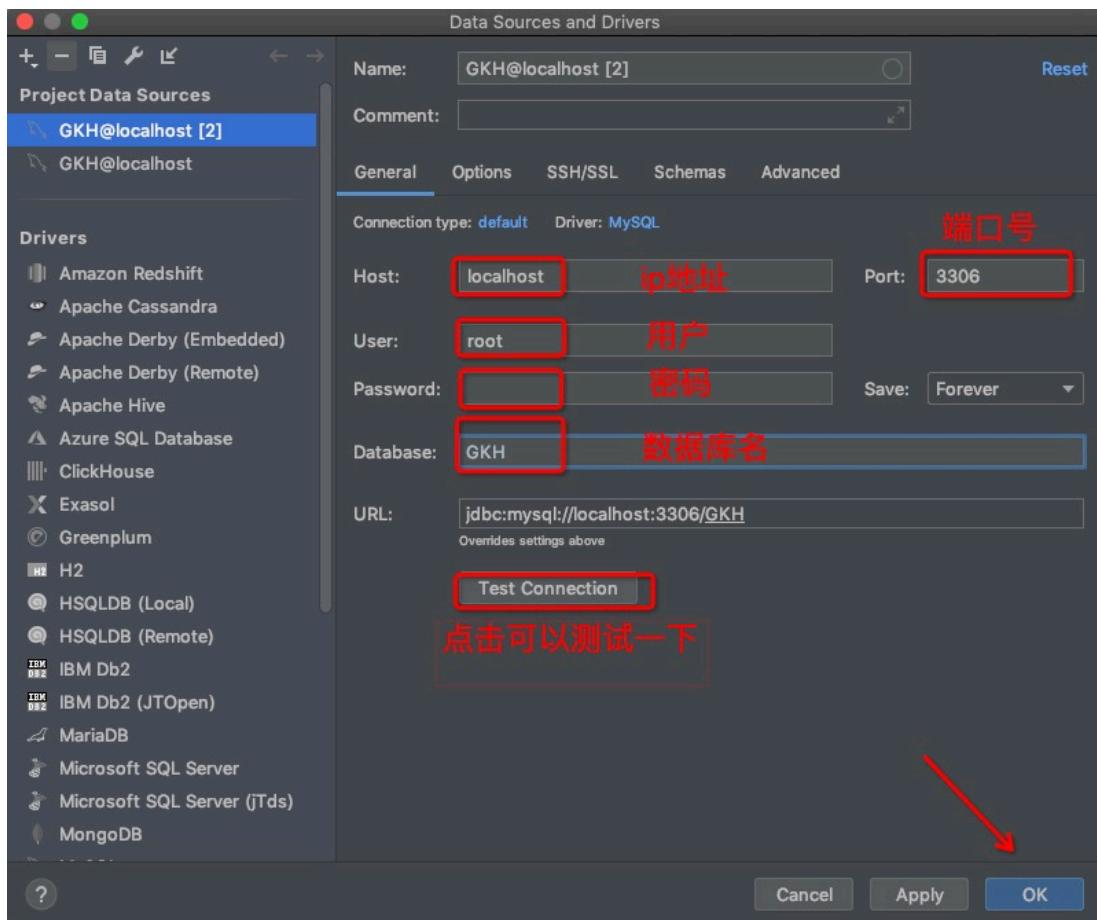
连接数据库 的方法

1. 使用Navicat连接。
2. 使用cmd 或者 terminal连接。
3. 使用pycharm 连接。

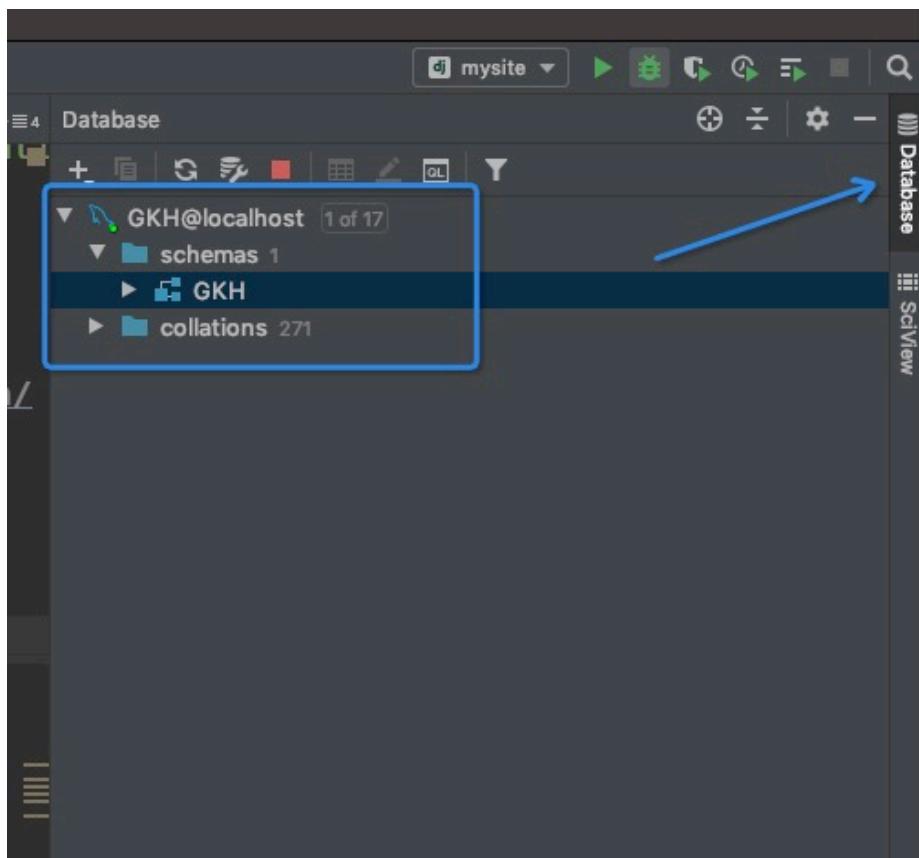
pycharm编辑器连接。

在编辑器的 右上角 点一下， 选择 Data Source -- Mysql。

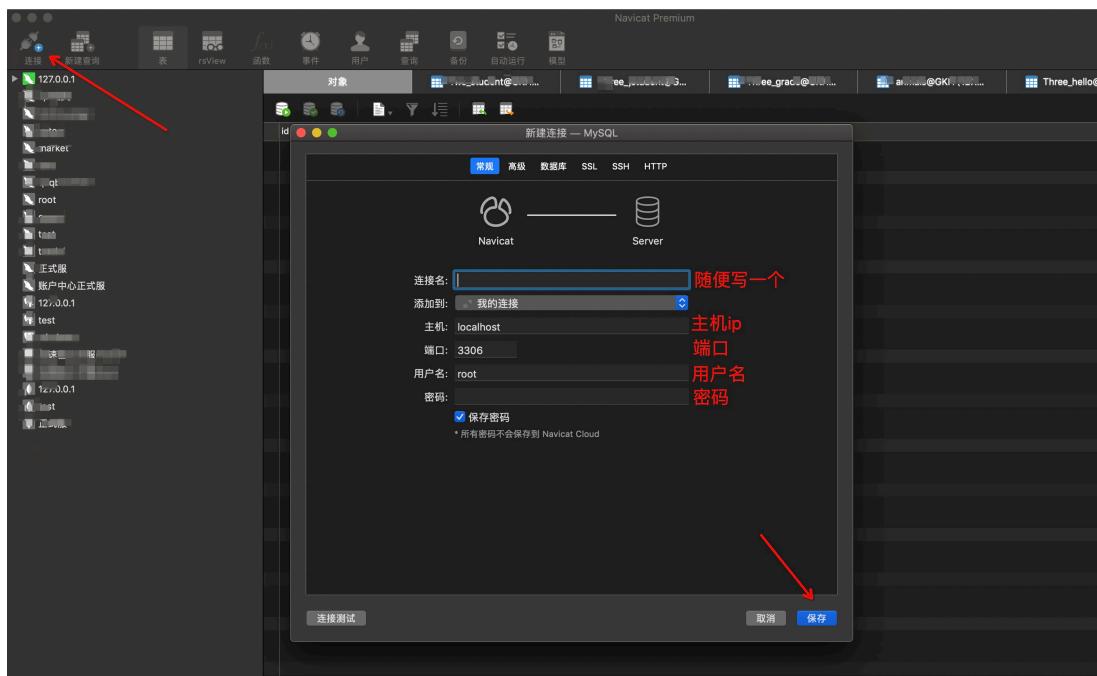




连接成功。



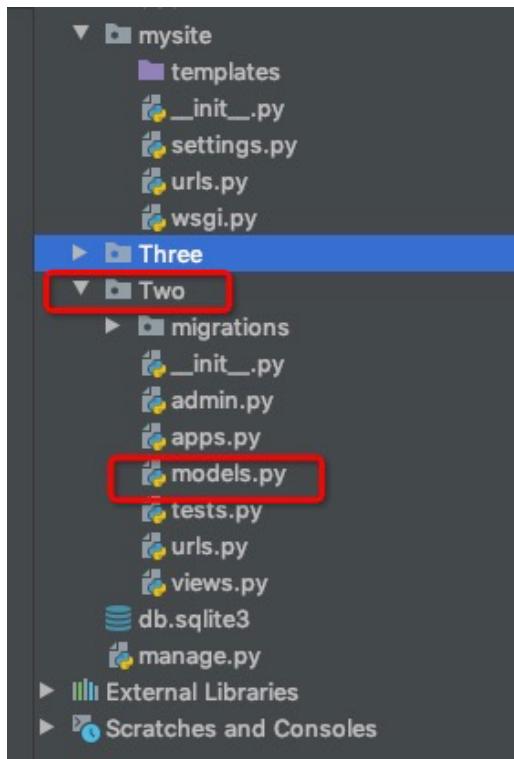
2. Navicat连接。



ORM

- 1 什么是ORM?
- 2 ORM (对象关系映射) 指用面向对象的方法处理数据库中的创建表以及数据的增删改
- 3 简而言之, 就是将数据库的一张表当作一个类, 数据库中的每一条记录当作一个对象
- 4 在Django中定义一个类, 就是在数据库中创建一张表格。在Django中实例化一个对象
- 5 就是在数据库中增加了一条记录。在Django中删除一个对象, 就是在数据库中删除
- 6 在Django中更改一个对象的属性, 就是在数据库中修改一条记录的值。
- 7 在django中遍历查询对象的属性值, 就是在数据库中查询记录的值。

打开 新建App 的 目录 里的 mysqls.py 我的是 Two 。



```
1 from django.db import models
2
3 class City(models.Model):
4     id = models.AutoField(primary_key=True) #id 主键
5     name = models.CharField(max_length=25, db_column='name')
6
7
8 class LOL(models.Model):
```

```
9      # p_id = models.AutoField(primary_key=True, db_column='id')
10     p_name = models.CharField(max_length=32, unique=True, db_
11     p_age = models.IntegerField(default=18, db_column='age')
12     p_sex = models.BooleanField(default=False, db_column='sex'
13
14     class Meta:
15         db_table = 'lol'    #表的名字
```

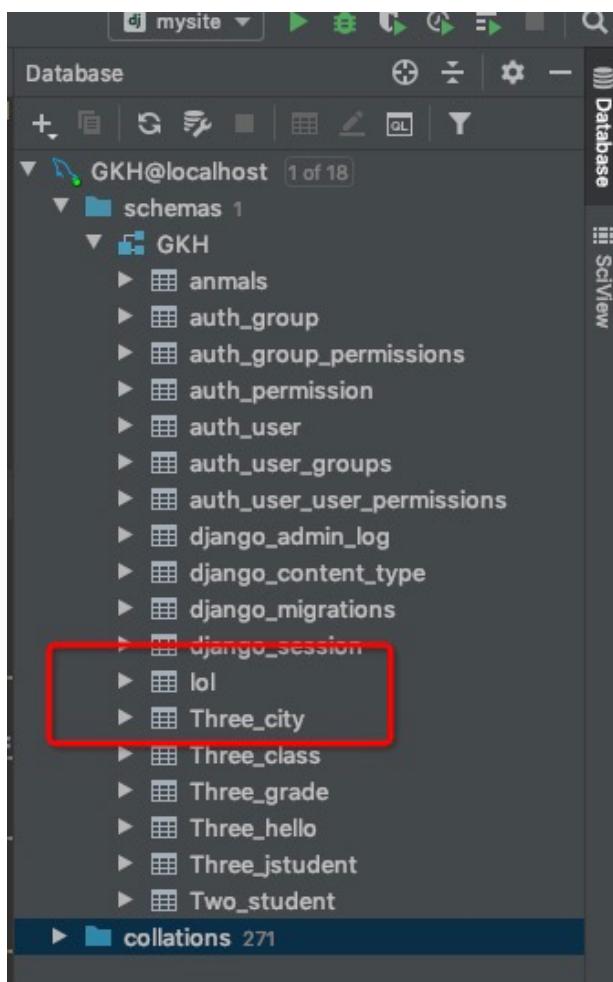
执行命令 生成表。

```
1 (venv) ➔ mysite python manage.py makemigrations  #执行命令
2 Migrations for 'Three':
3     Three/migrations/0003_city_lol.py
4         - Create model City
5         - Create model LOL
6 (venv) ➔ mysite
7
```

```
1 (venv) ➔ mysite python manage.py migrate   #执行命令
2 Operations to perform:
3     Apply all migrations: Three, Two, admin, auth, contenttypes,
4 Running migrations:
5     Applying Three.0003_city_lol... OK
6 (venv) ➔ mysite
7
```

数据库里 已经生成了 city 表 和 lol 表。

需要注意的是：有可能 django的版本和 pymysql的版本不匹配就会报错。



如果插入失败，报错。可能是版本的问题。需要改一下源代码。

```

25 # Some of these import MySQLdb, so import them after checking if it's installed.
26 from .client import DatabaseClient          # isort:skip
27 from .creation import DatabaseCreation      # isort:skip
28 from .features import DatabaseFeatures       # isort:skip
29 from .introspection import DatabaseIntrospection # isort:skip
30 from .operations import DatabaseOperations   # isort:skip
31 from .schema import DatabaseSchemaEditor     # isort:skip
32 from .validation import DatabaseValidation    # isort:skip
33
34
35 version = Database.version_info
36 #if version < (1, 3, 13):
37 #    raise ImproperlyConfigured('mysqlclient 1.3.13 or newer is required; you have %s.' % version)
38
39
40 # MySQLdb returns TIME columns as timedelta -- they are more like timedelta in
41 # terms of actual behavior as they are signed and include days -- and Django
42 # expects time.
43 django_conversions = {
44     **conversions,
45     **{FIELD_TYPE.TIME: backend_utils.typecast_time},
46 }
47

```

The code editor shows the Django MySQL backend code. A red box highlights the 'base.py' file and a specific line of code: '#if version < (1, 3, 13):'. This line is part of a conditional block that checks the MySQL client version and raises an error if it's too old.

在重新生成表，应该就可以了。

1.打开 Two中的 urls文件。

```
1 from django.conf.urls import url
2
3 from Two import views
4
5 urlpatterns = [
6     url(r'^home/$', views.index),
7
8     url(r'^addhero/$', views.addhero), #添加 路由请求 url
9 ]
```

2.在打开 Two中的 views 文件。

```
1 from Three.models import LOL
2 from django.http import HttpResponseRedirect
3
4 #数据库增删改查。
5
6 def addhero(request):
7     #插入数据的第一种方法
8     # a=LOL.objects.create(p_name='瑞雯')
9     # a.save()
10
11     # 插入数据的第二种方法
12     p = LOL()
13     p.p_name='伊泽瑞尔'
14     p.p_age='18'
15     p.save()
16     return HttpResponseRedirect("添加成功")
17
18
19
```

3.然后打开 浏览器 或者 postman， 使用curl 命令也行。

输入 127.0.0.1:8000/addhero

1.数据库会插入两条数据。 (增删改查) ===插入数据

The screenshot shows a MySQL Workbench interface with a database named 'GKH'. A table named 'lol' is selected. The table has four columns: 'id', 'name', 'age', and 'sex'. There are two rows of data:

	id	name	age	sex
1	1	瑞雯	18	0
2	2	伊泽瑞尔	18	0

2.查询 数据

```
1
2 def addhero(request):
3     #查询所有数据
4     p = LOL.objects.all()
5     for s in p:
6         print(s.p_name)
7         print(s.p_age)
8     return HttpResponse("200")
9
10    #查询 id = 1
11    p= LOL.objects.filter(id=1)
12    for s in p:
13        print(s.p_name)
14        print(s.p_age)
15    #通过 名字 查询
```

```
16     p= LOL.objects.filter(p_name='瑞雯')
17     for s in p:
18         print(s.p_name)
19         print(s.p_age)
20
21     #查询    1< id <10 的 数据
22     #gte 大于等于    lte 小于等于
23     #gt  大于      lt  小于
24     p = LOL.objects.filter( id__lt = 10, id__gt = 1)
25     for s in p:
26         print(s.p_name)
27         print(s.p_age)
28
29     #查新 id =1  id=5,id=11 的数据
30     p = LOL.objects.filter( id__in = [1,5,11])
31     for s in p:
32         print(s.p_name)
33         print(s.p_age)
34     # 查询 包含 某一个 内容的 数据
35     #比如 瑞雯    伊泽瑞尔    都有 瑞 这个相同内容
36     # contains 和 icontains 区别
37     # icontains 大小写不敏感
38     p = LOL.objects.filter(p_name__contains="瑞")
39     for s in p:
40         print(s.p_name)
41         print(s.p_age)
42     ######
43     瑞雯    伊泽瑞尔
44     18        18
45     #####
46
47     #查找 某一数据的 范围  相当于SQL语句中的 bettwen and
48     #在 两个数 之间
49     p = LOL.objects.filter(id__range=[1, 4])
```

```
50     for s in p:
51         print(s.p_name)
52         print(s.p_age)
53
54     # __exact      精确等于 like 'aaa'
55     #__iexact      精确等于 忽略大小写 ilike 'aaa'
56     p = LOL.objects.filter(p_name__exact="瑞雯")
57     for s in p:
58         print(s.p_name)
59         print(s.p_age)
60
61     # 判空
62     #判断    p_name   用户名是否为空
63     p = LOL.objects.filter(p_name__isnull=True)
64
65     for s in p:
66         print(s.p_name)
67         print(s.p_age)
68
69
70     #查询结果 以 map 形式 返回
71     p = LOL.objects.filter(p_name='亚索').values()
72     print(p)
73
74
75     #还有其他查询
76     # get():          返回与所给筛选条件相匹配的对象
77
78     # count():        返回数据库中匹配查询(QuerySet)的对象数
79
80     # first():        返回第一条记录
81
82     # last():         返回最后一条记录
83
```

```
84     # exists():          如果QuerySet包含数据，就返回True，否则
85
86     # reverse():         对查询结果反向排序
87
88     # distinct():        从返回结果中剔除重复纪录
89
90     # reverse():         对查询结果反向排序
91
92     # order_by(*field): 对查询结果排序
```

3.更新数据

```
1
2 #更新数据，字段信息。
3 #把id=4的数据 name 更新为 VN
4 p = LOL.objects.filter(id=4).update(p_name='VN')
5
6 #更新 name=VN age=3 sex=0
7 p = LOL.objects.filter(id=4).update(p_name='VN',p_age='3',p_se
8
9
```

4.删除数据

```
1 #删除 id=4 的数据
2 p = LOL.objects.filter(id=4).delete()
3 print(p)
```

Django Orm 里的一对一，一对多，多对多。

一对一

```
1 一对一的表，两表的属性实际上完全可以合并成一个表，共用一个主键即可。
2 班级 和 班长 一对一           中国居民 和 身份证号 一对。
3 OneToOneField: 一对一
```

一对多，多对一。

- 1 一张表中的数据 对应 另一张表中 多个数据，就是一对多关系。
- 2 班级 和 学生 一个班级有很多学生 很多学生 对应一个班级。
- 3 在多的一方的表里面，添加外键
- 4 ForeignKey: 一对多
- 5 ForeignKey: 多对一

多对多

- 1 多对多：一个班级有若干个老师，一个老师也可以带若干个班级。
- 2 一门课程同时有若干个学生选修，一个学生可以同时选修多门课程。
- 3 ManyToManyField: 多对多

代码一对一。

- 1 一对一 一个人 对应 一个生日。 (人 对应 出生日期)
- 2 新建 两张表
- 3 Person表 存放 人名
- 4 Birthday表 存放 生日
- 5

```
1 #Person表
2 class Person(models.Model):
3     name = models.CharField(max_length=10, db_column='name')
4     #重新命名生成的表名 为 person
5     class Meta:
6         db_table = 'person'
7 # 生日表
8 class Birthday(models.Model):
9     # 与Person表为一对一， Person表为主表
10    person = models.OneToOneField("Person", on_delete=models.CASCADE)
11    birthday = models.CharField(max_length=10, db_column='birthday')
12    #重新命名生成的表名 为 birthday
13    class Meta:
14        db_table = 'birthday'
```

使用命令 创建表。

```
1 python manage.py makemigrations
2 python manage.py migrate
```

数据库就生成新创建的两张表

可以手动插入数据，也可以直接从数据库插入数据。

```
1 主表 Person 子表 Birthday
2 子表查询主表,找到生日 对应的人名    返回 '瑞雯'。
3 #写法一
4 "子表对象.主表表名的小写.主表字段名"
5 通过Birthday表查到为 生日为 "2020-5-20", 查找到对应 姓名
6 print(Birthday.objects.get(birthday="2020-5-20").person.name)
7
8 #写法二
9 "主表.objects.get(子表名小写__子表字段='xxx').主表字段名"
10 反从主表查询
11 print(Person.objects.get(birthday__birthday="2020-5-20").name)
12
13
14 主表查询子表 找到 瑞雯 的 生日    返回 '2020-05-20'
15
16 #写法一
17 '主表对象.子表表名的小写.子表字段名'
18 print(Person.objects.get(name="瑞雯").birthday.birthday)
19
20
21 #写法二
22 '"子表.objects.get(一对一的子表字段__主表字段="xxx").子表字段"'
23 print(Birthday.objects.get(person_name="瑞雯").birthday)
24
25
```

一对多 插入数据

```
1 先插入 人名 并添加 生日
2 先在主表中创建颜色，并实例化给颜色表对象
3 p=Person.objects.create(name="亚索")
4 更新Birthday表，birthday字段为Person表对象，添加birthday字段
5 Birthday.objects.create(person=p,birthday="2020-5-21")
6
7 #方法二
8 p=Person.objects.create(name="卡特琳娜")
9 b=Birthday(person=p,birthday="2020-5-22")
10 b.save()
11
12 #方法三
13 p=Person.objects.create(name="艾瑞莉娅")
14 b={'birthday':'2020-5-23'}
15 Birthday.objects.create(person=p,**b)
16
```

更新数据

```
1 把亚索的生日 2020-5-20 改为 2021-5-20
2 p = Person.objects.get(name="亚索")
3 #写法一
4 Birthday.objects.filter(birthday="2020-5-21").update(person=p,
5 #写法二
6 把亚索改成德玛
7 p = Person.objects.get(name="亚索")
8 p.name="德玛"
9 p.save()
10
```

删除数据

```
1 删除生日为 2020-5-20 的数据
2 Birthday.objects.get(birthday="2020-5-20").delete()
```

```
3 删除 名字 为 瑞雯的数据
4 Person.objects.filter(name="瑞雯").delete()
5 注意：如果删除主表中的数据，子表的数据 会一起删除。
6
```

一对多（外键）

```
1 一个人 有多个 爱好      骑车 跑步 听歌
2 新建 一个 Hobby表
3 class Hobby(models.Model):
4     personid = models.ForeignKey("Person", on_delete=models.CASCADE)
5
6     hobby = models.CharField(max_length=10)
7     class Meta:
8         db_table = 'hobby'
9
10
```

生成表

```
1 python manage.py makemigrations
2 python manage.py migrate
```

先往hobby表 插入一些数据

Person表

```
1 id    name
2 1    瑞雯
3 2    亚索
4 3    德邦
```

Birthday表

```
1 id    birthday   personid
2 1    2020-5-20    1
3 2    2020-5-21    2
4 3    2020-9-30    3
```

```
1 id    hobby       person_id
```

2	1	骑车	1
3	2	跑步	2
4	3	听歌	1
5	4	看电影	1
6	5	睡觉	2
7	6	旅游	3

```

1 外键表联合查询:
2
3 外键子表查询母表,与一对子表查询母表形式一致
4 找到 '骑车' 所属的 'Hooby表' 中的爱好--返回: '瑞雯'
5 写法1:
6 通过子表查询母表, 写法: "子表对象.主表表名的小写.主表字段名" ;
7 通过Hobby表查到hobby为"骑车", 查找到对应 name
8
9 print(Hobby.objects.get(hobby="骑车").person.name)
10
11 主表是 Person表
12 写法2, 反向从主表入手:
13 返回'瑞雯', 通过子表查询主表, 但形式上是从主表对象自身直接获取字段,
14 写法: "主表.objects.get(子表名小写__子表字段="xxx").主表字段名" ;
15
16 print(Person.objects.get(hobby__hobby="骑车").name)
17
18
19 外键主表查询子表,与一对一形式不同, 因为子表为"多", 不能像一对一 一样通过
20 .get().子表.子表字段的方式获取, 但与多对多主表查询子表一致
21 找到'瑞雯'的所有爱好--返回: [<Hobby: 骑车>, <Hobbys: 看电影>, <Hobby: 听歌>]
22
23 person=Person.objects.get(name="瑞雯")
24 hobby=person.hobby_set.all()
25 for h in hobby:
26     print(h.hobby)
27

```

```
28 增加数据
29 Hobby.objects.create(person=models.Person.objects.get(name="瑞雯"))
30 更新数据
31 #全部更新
32 Person.objects.filter(person__name="瑞雯").update(hobby="弹钢琴")
33
34 #更新指定数据 字段
35
36 obj = Person.objects.get(name="瑞雯")
37 obj.hobby_set.filter(hobby='跑步').update(hobby="游泳")
38
39
40 #删除数据
41 #hobby 表中有 2条数据 hobby 字段 为 跳舞
42 #这个只能删除一条 数据 如果 查询到多条 会报错。
43
44
45 Person.objects.get(name="瑞雯").delete()
46 #删除 hobby表中的 数据
47 Hobby.objects.get(hobby="跳舞").delete()
48
49 -----
50 #都删除了。 hobby 表中有 2条数据 hobby 字段 为 跳舞
51
52 Hobby.objects.filter(hobby="跳舞").delete()
53
54
55 #删除主表中的数据 子表中的数据 会一起删除。
56 如果删除了 瑞雯 则会把 hobby表中的 所有爱好 全部删除。
57
58 Person.objects.filter(name='瑞雯').delete()
59
```

多对多

```
1 差不多~~~~~  
2
```

Django 接收 前端参数

1.postman请求不通，设置一下。

```
1 postman 请求 会报错  
2  
3 1.在 settings.py文件中 加上 APPEND_SLASH = False ( 如果有就 改为  
4 2.在 settings.py文件中 , 找到  
5  
6  
7 MIDDLEWARE = [  
8     'django.middleware.security.SecurityMiddleware',  
9     'django.contrib.sessions.middleware.SessionMiddleware',  
10    'django.middleware.common.CommonMiddleware',  
11    #'django.middleware.csrf.CsrfViewMiddleware',    注释  
12    'django.contrib.auth.middleware.AuthenticationMiddleware'  
13    'django.contrib.messages.middleware.MessageMiddleware',  
14    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
15 ]  
16 把上面的    这个注释掉    'django.middleware.csrf.CsrfViewMiddleware'  
17  
18 这样就可以请求了。
```

2.好像前端请求的时候 要在 url路径后面 加上 /

比如 请求为： http:127.0.0.1:8000/hello

后面要加上 / http:127.0.0.1:8000/hello/

不是很清楚，先记下来。

3 POST 请求

```
1 def hello(request):
```

```
2
3     if request.method == 'POST':
4         #表单提交参数    获取    username
5         username = request.POST["username"]
6         #表单提交参数    获取    pwd
7         pwd = request.POST.get("pwd")
8         print(username)
9         print(pwd)
10
11        #获取request.body    获取非表单数据,如json
12        #request.body返回的是一个byte的对象
13        s=request.body
14        print(s)
15
16        #####
17        这是postman    post    请求    json
18        {
19            "id":1,
20            "name":"瑞雯"
21        }
22        #####
23        id = eval(request.body.decode()).get('id')
24        name = eval(request.body.decode()).get('name')
25        print('name=',name)
26        print("data=",id)
27
28        #获取    请求路径
29        path = request.path
30        print('path=',path)
31        #获取    请求方法
32        method = request.method
33        print("method",method)
34
35    if request.method == 'GET':
```

```
36      #获取 get 请求 参数  
37      username = request.GET.get("username")  
38  
39      print(username)  
40      return HttpResponse("200")
```
