基于Flask的web项目开发实验一

# Python环境准备

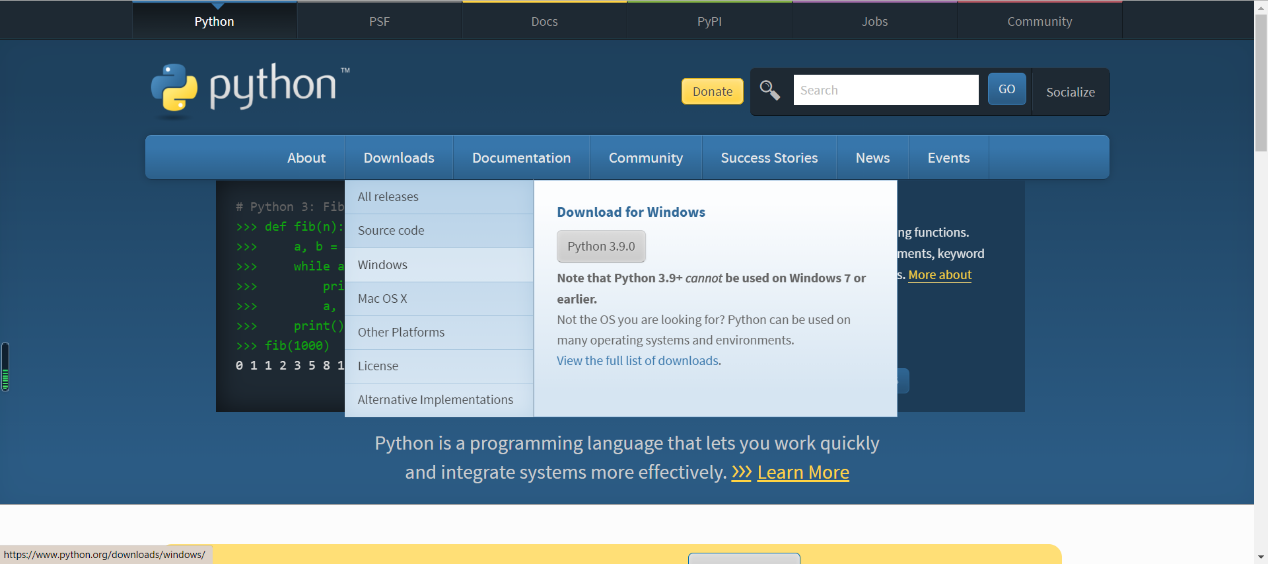
在开始flask项目前，需要安装Python环境和PyCharm。

请未安装的同学直接从官网下载安装。

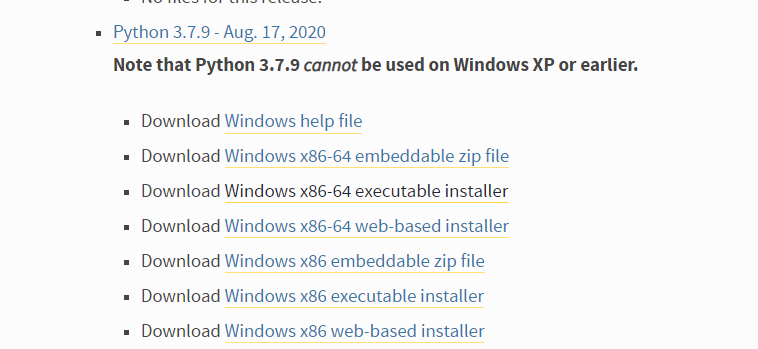
## Python的安装和配置

Python官网：<https://www.python.org/>

进入Python官网后选择DOWNLOAD选项下的Window选项。

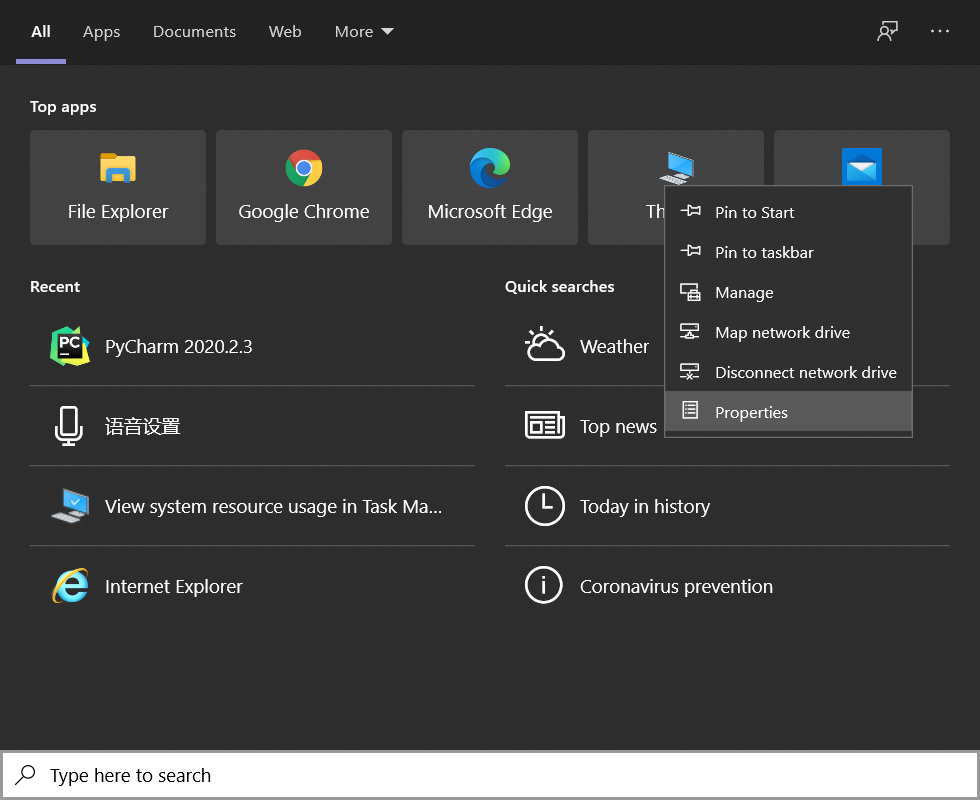


在Window页面中选择Python3.7版本或3.8版本的安装，Python3.9版本刚发行，版本代码中存在着很多bug，还不够稳定，不建议大家使用。

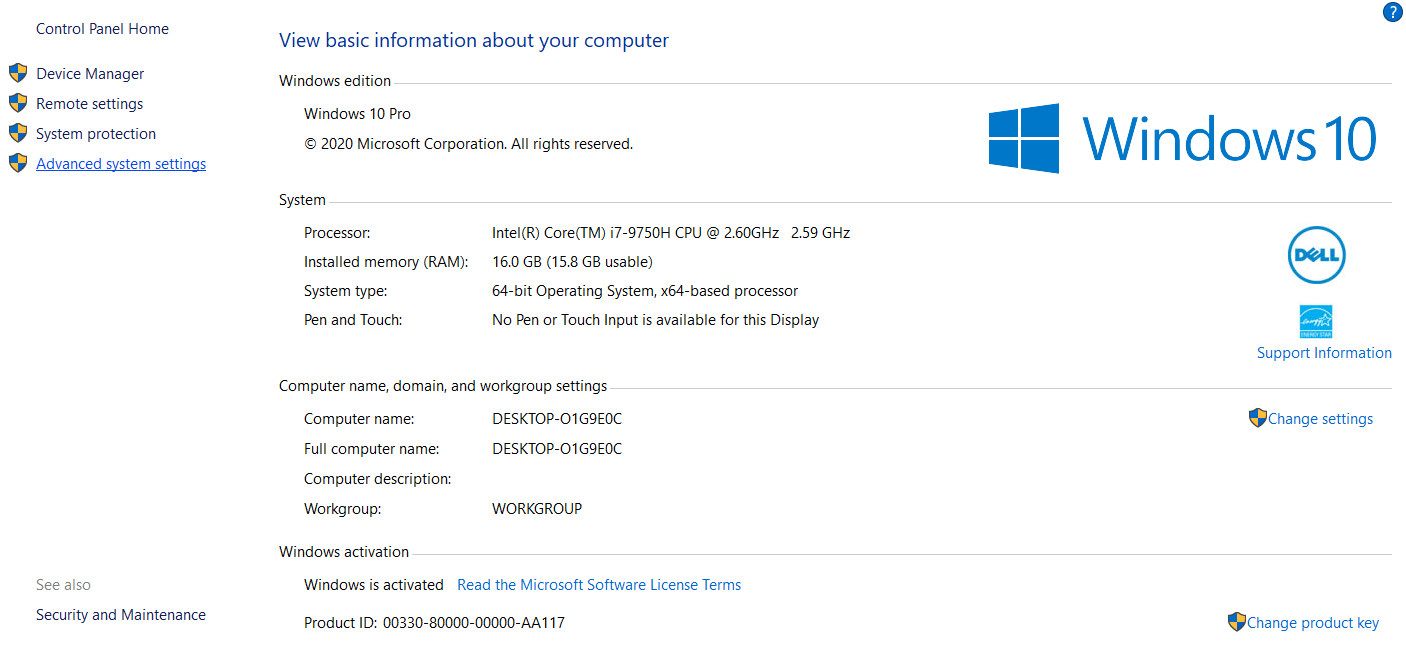


下载完成后只需要按照步骤安装即可，安装过程中注意选择添加环境变量，如果忘记添加环境变量可安装完成后自行添加。

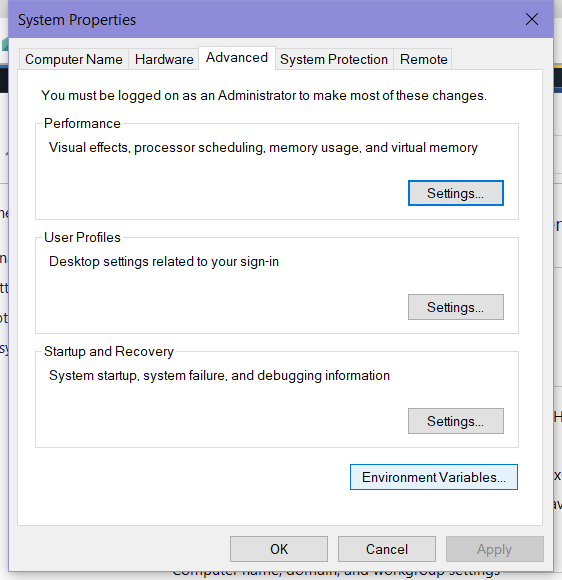
自行添加环境变量：右键“我的电脑”选择“属性”



在打开的页面选择高级系统设置。

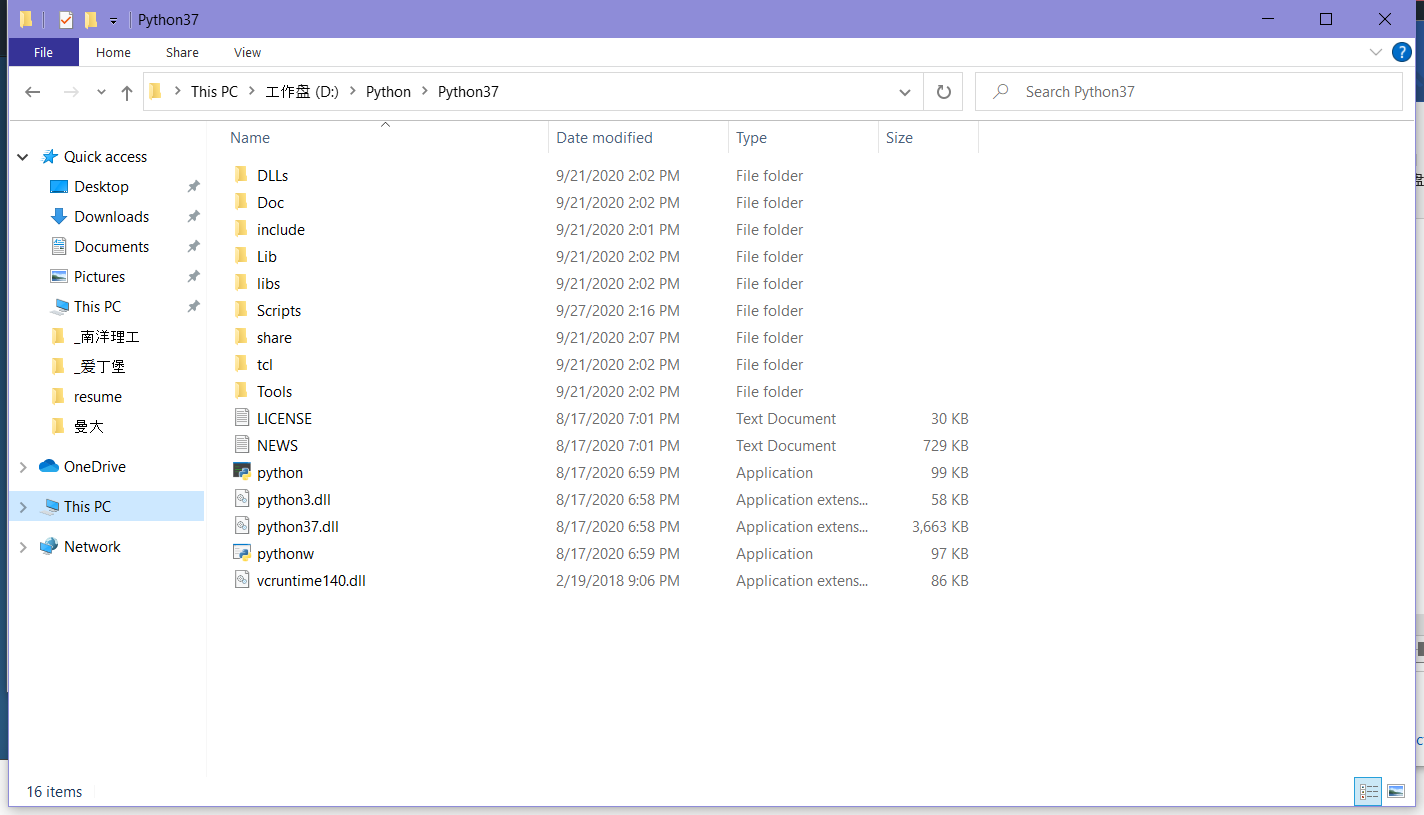


之后点击右下角的“环境变量”按钮。

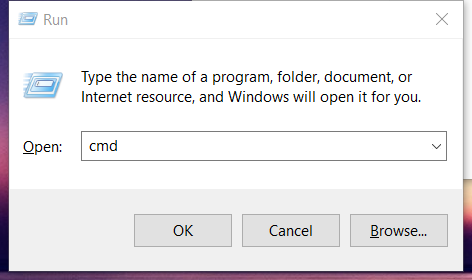


确保“系统变量”的PATH中含有Python安装位置和其目录下Scripts文件的路径即可。

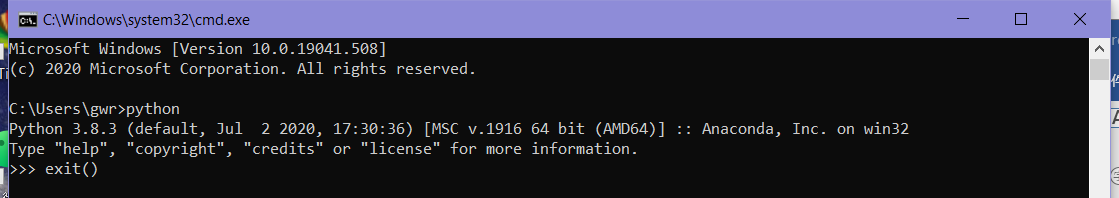




环境配置完成后使用“win+R”快捷键打开cmd。

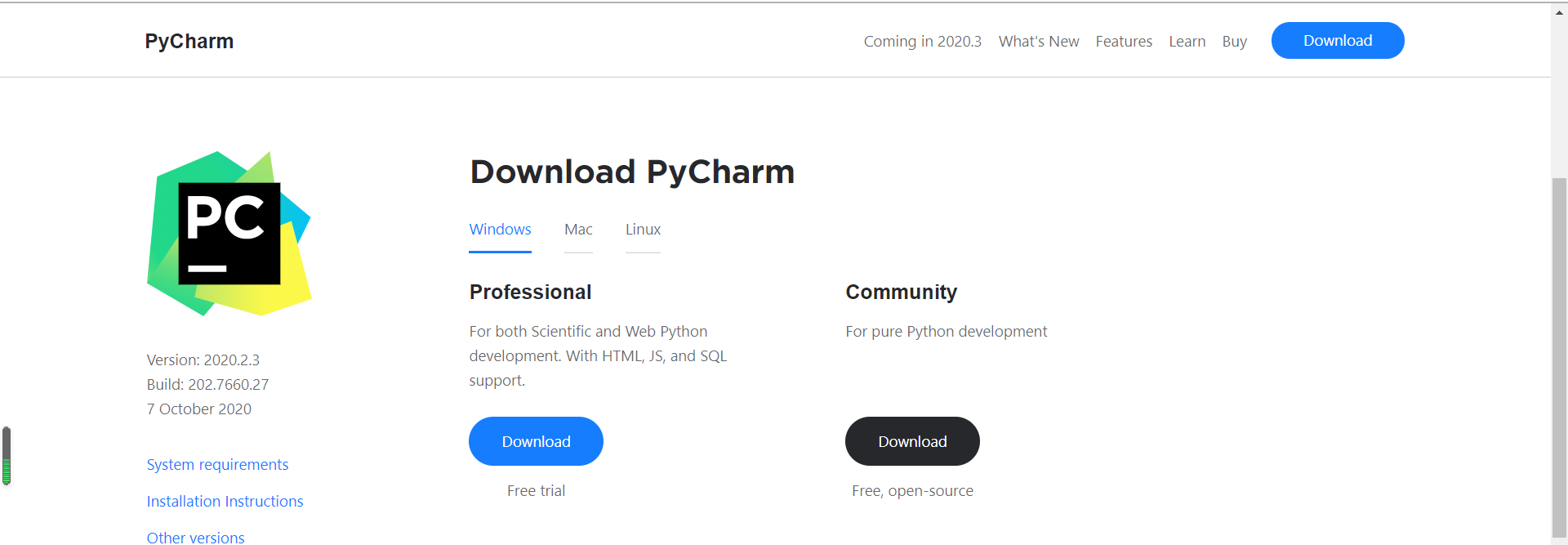


在命令行中输入“python”，如果显示了版本号即表明安装成功。



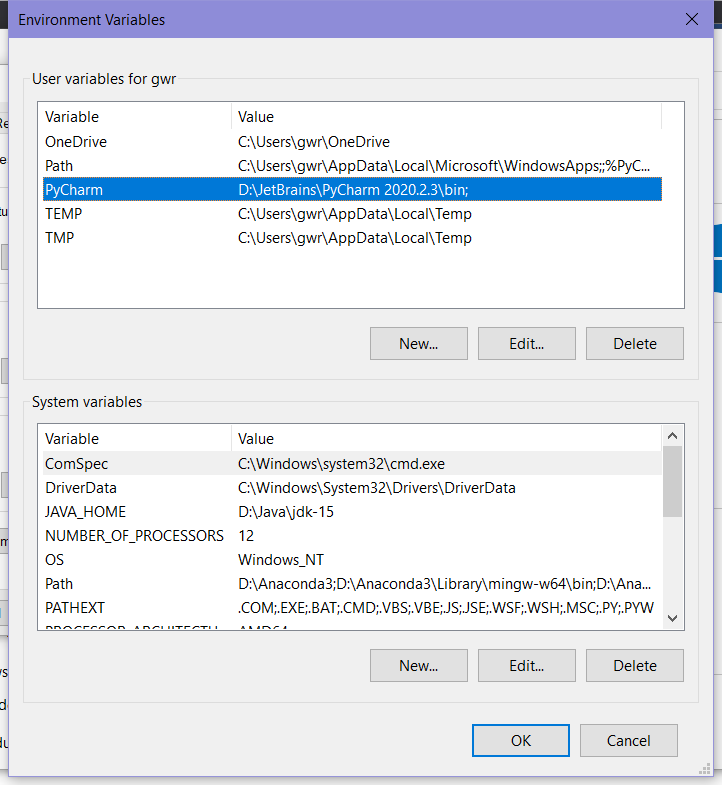
## PyCharm的安装和激活

PyCharm官网：<https://www.jetbrains.com/pycharm/download/#section=windows>



从官网介绍中可以看到专业版PyCharm支持更多功能，也包括了我们要用的Flask，然而社区办PyCharm没包含这些功能，只能运行基础Python项目，所以我们选择专业版进行下载，也就是点击上图的蓝色按钮。

下载过程中注意选择添加路径，从环境变量中可以看到添加的路径进行确认。



要注意JetBrains的软件基本都需要激活，这里我们选择永久激活，减少了每年一激活的麻烦。

安装PyCharm完成后，我们首先选择“Evaluate for free”直接打开免费试用版。New一个新项目进入PyCharm主界面。之后将激活文件的压缩包拉到主界面中。激活文件如下。

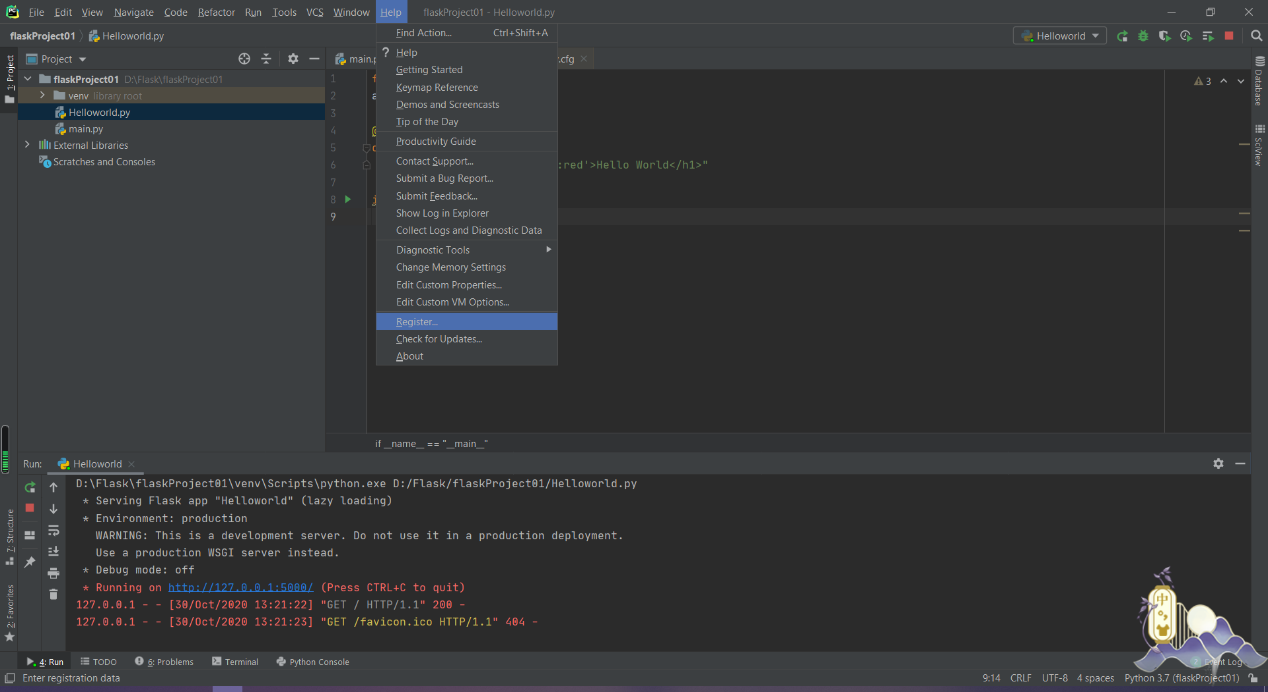


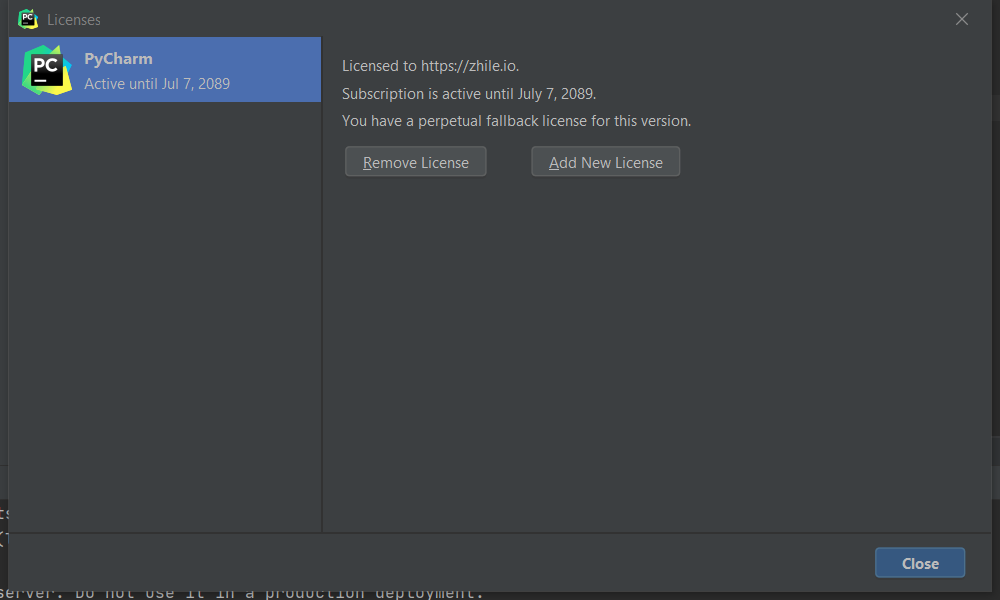
激活码和激活文件可从北航云盘中下载，下载地址为： https://bhpan.buaa.edu.cn:443/link/E5075171E91499DEB88D415AE10574FB

系统弹窗选择Restart，这之后会弹出新的弹窗，输入文件中的安装参数：

HbVMaxqQrTt4UvDUk8fhyFT7uRdv48bqJsQMwVFp53YH9PZC9l+ZxAgnA60cLwwhURw/e4DcZZtle4tQzur4yWRk35qghES4JLFlmKty/UNiYh6RZyXeCNPTCvIqxT9HR2YKqQT93be6AfLTcbJwPSLO201g+HbMbg6+aK1MUJI

选择安装后重启则生效。生效后可以通过查看注册时间确认自己是否激活成功，激活成功后注册时间会变为2089到期。

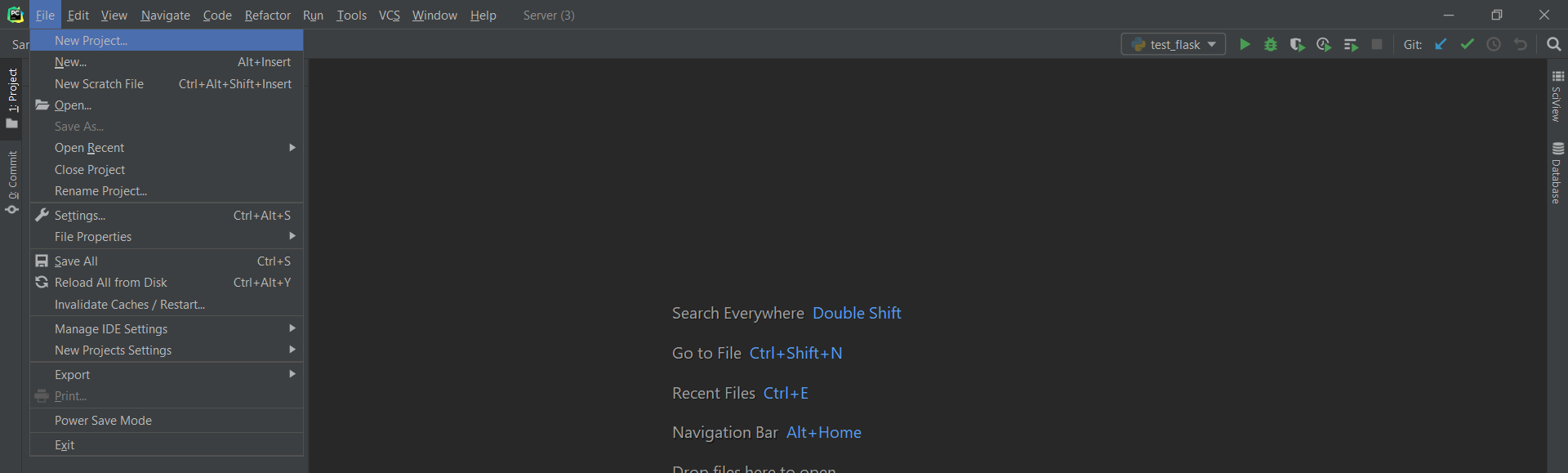




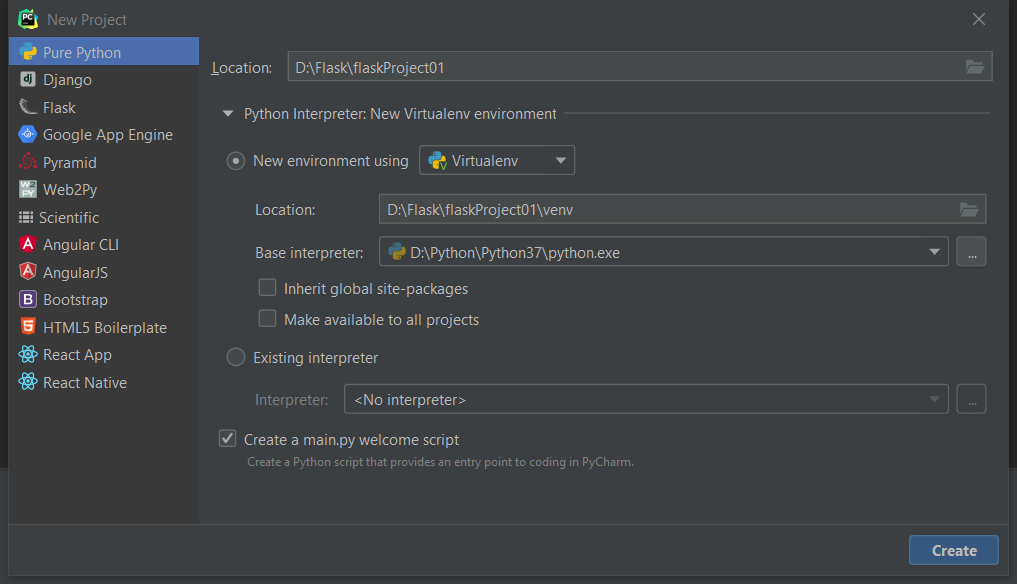
# 利用PyCharm创建Flask项目

## 创建Python项目

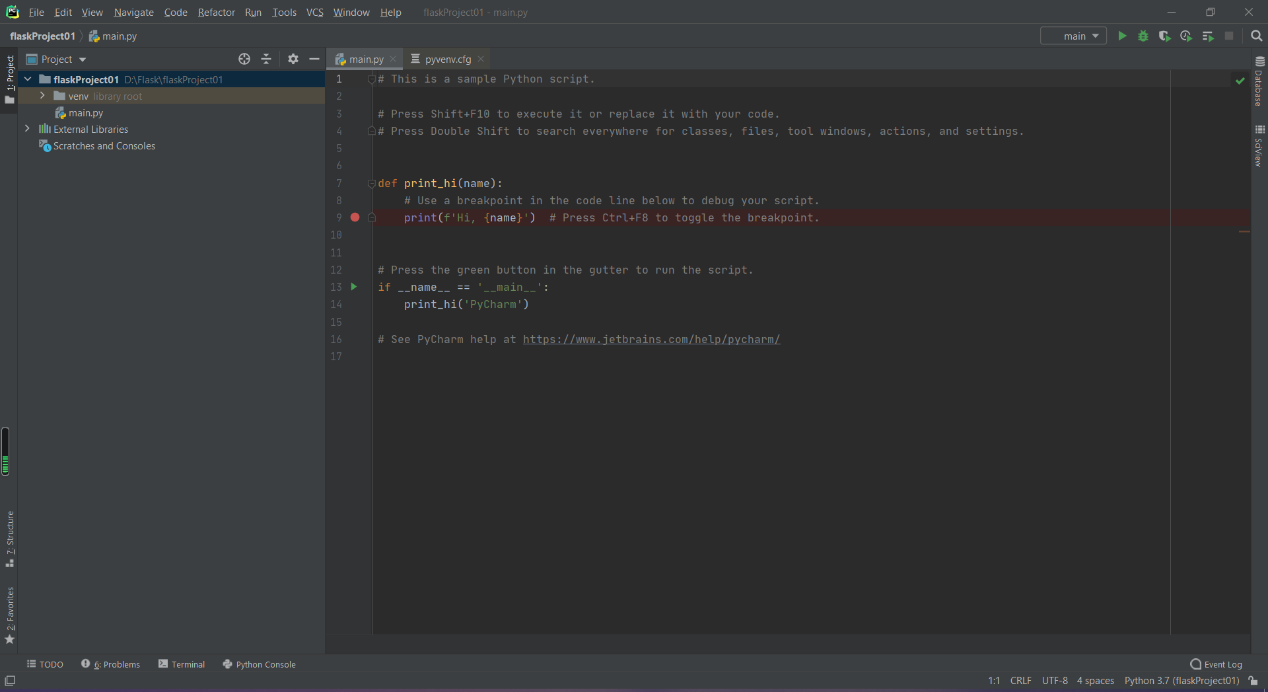
首先，我们来新建一个Python项目。



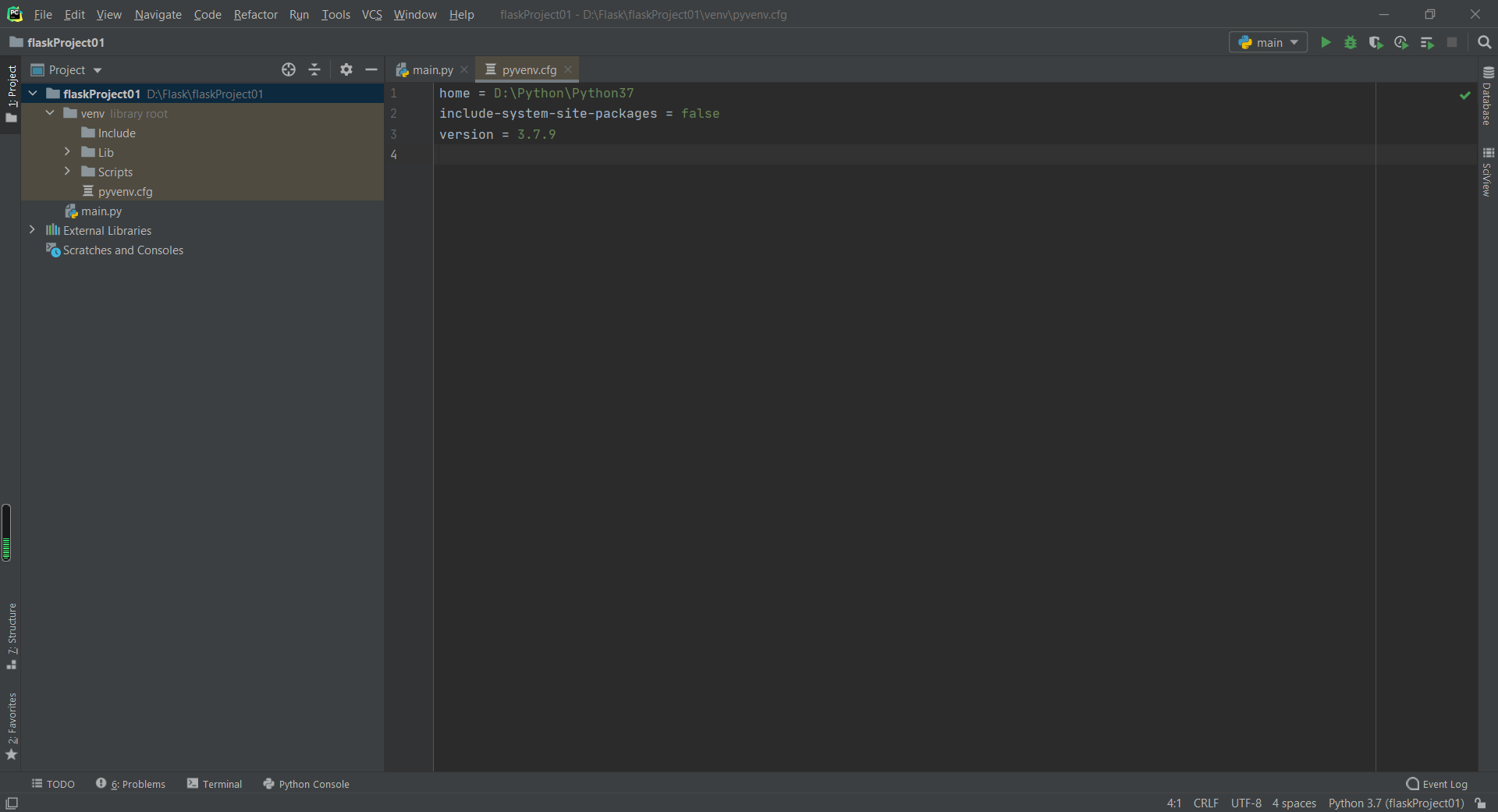
选择项目位置和虚拟化环境。



项目创建完成后包含了一个存储项目中虚拟化环境的“venv”文件和一个基础.py文件。

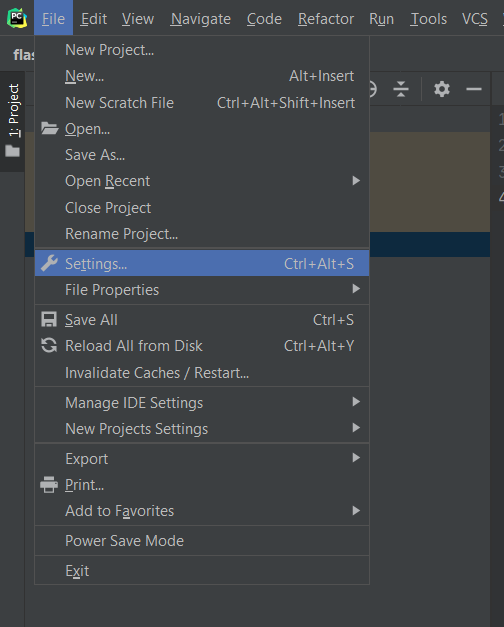


venv文件中记录了我们在这个项目开发过程中需要用到的一系列虚拟化环境，在多人协作开发过程中，每个人的虚拟化环境存储位置都是不同的，所以下载别的人python代码后，我们第一步要做的就是修改venv环境中pyvenv.cfg文件中的python路径和版本号，这样才可以引入自己的python环境，方便PyCharm自己为我们更新和安装所需的一系列配置。（要注意这种按项目存储的虚拟化环境venv文件是python3特有的，python2中不具备相关内容）

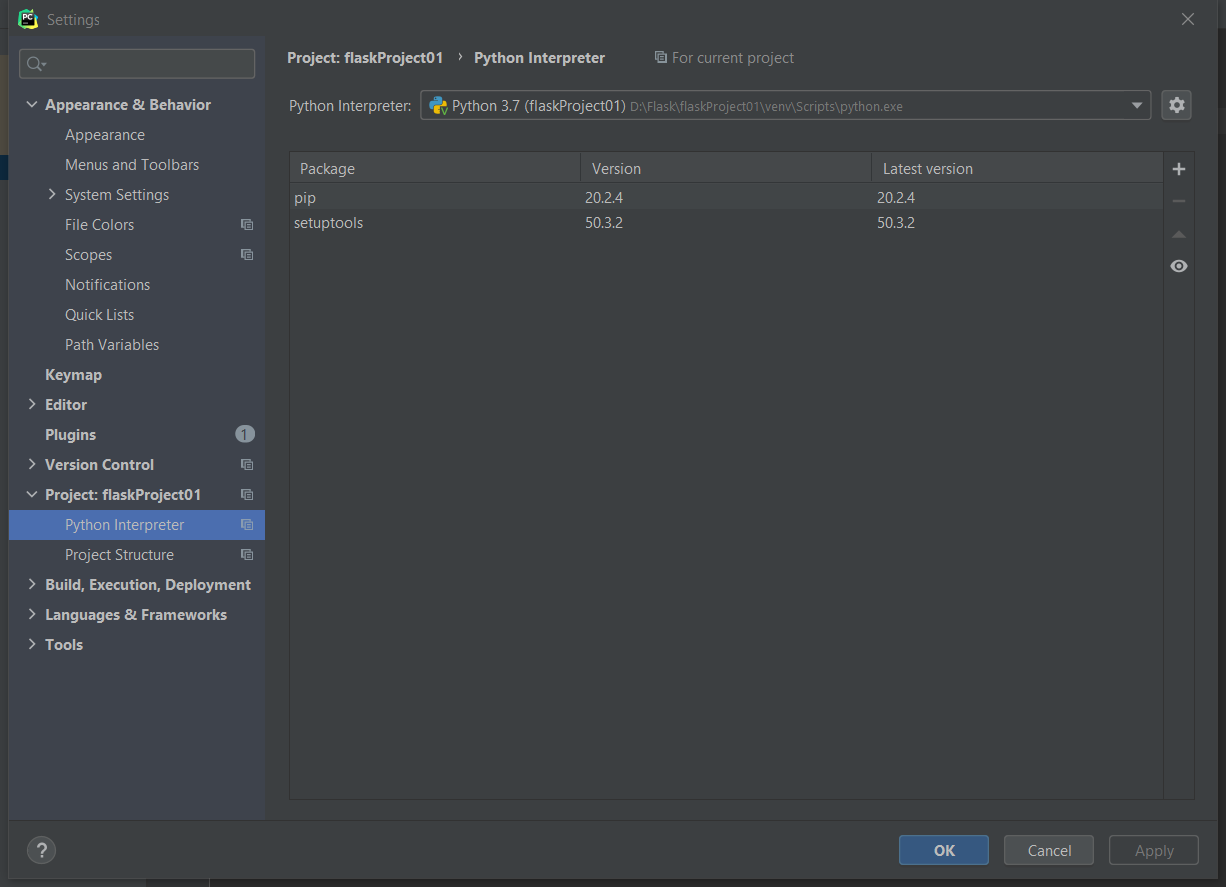


## 引入Flask环境

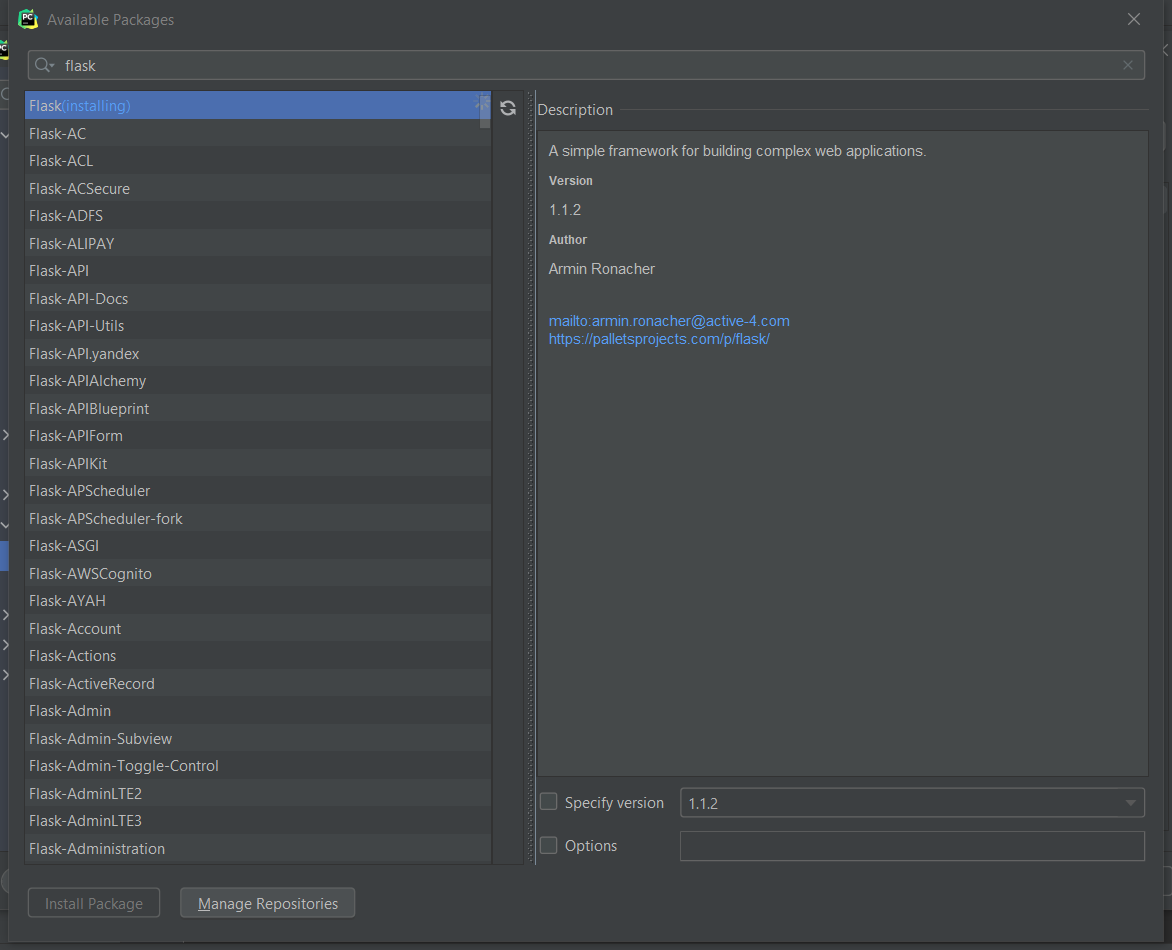
接下来是Flask环境的引入，选择设置选项。



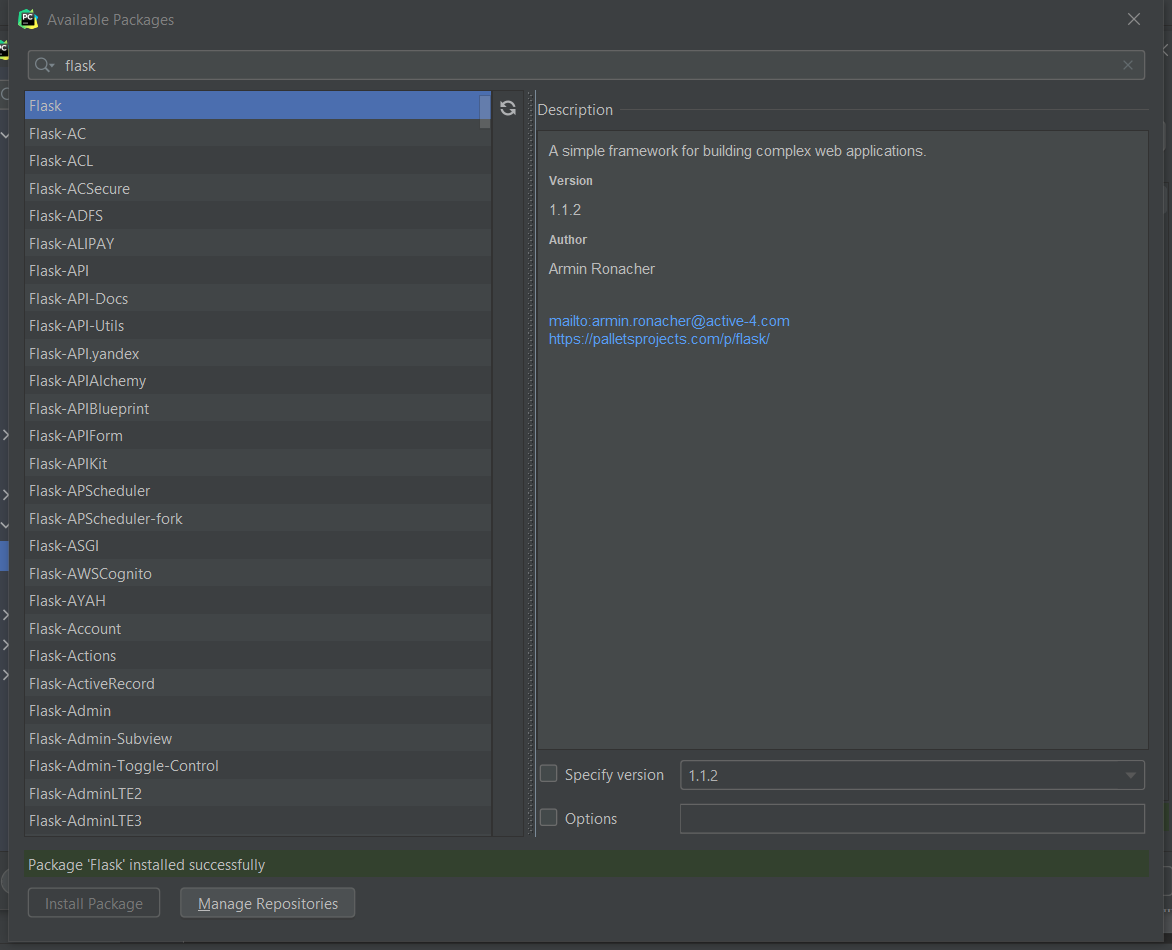
选择设置中的项目python解释器，可以看到此时的安装包中只有pip和setuptools，点击右上角的＋添加新的安装包。

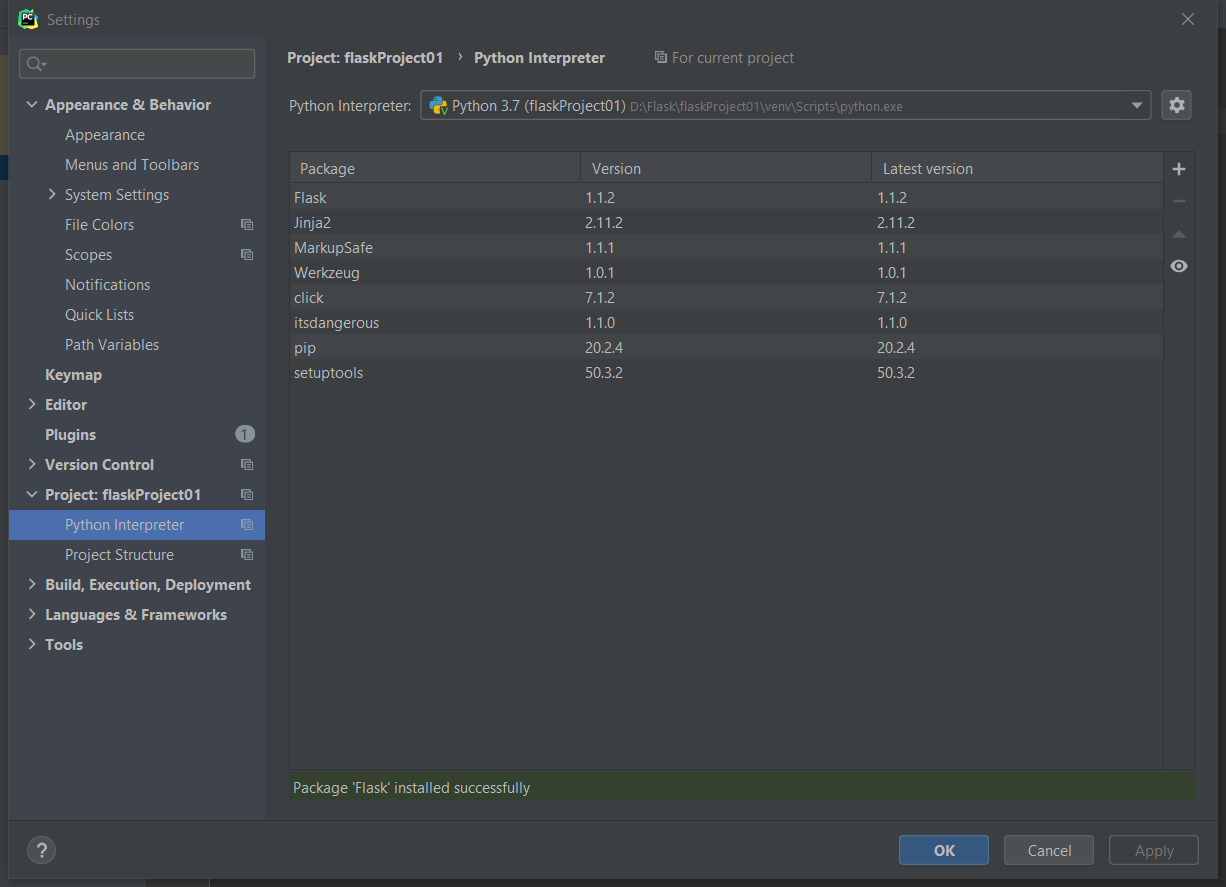


直接搜索flask，点击install package进行安装。

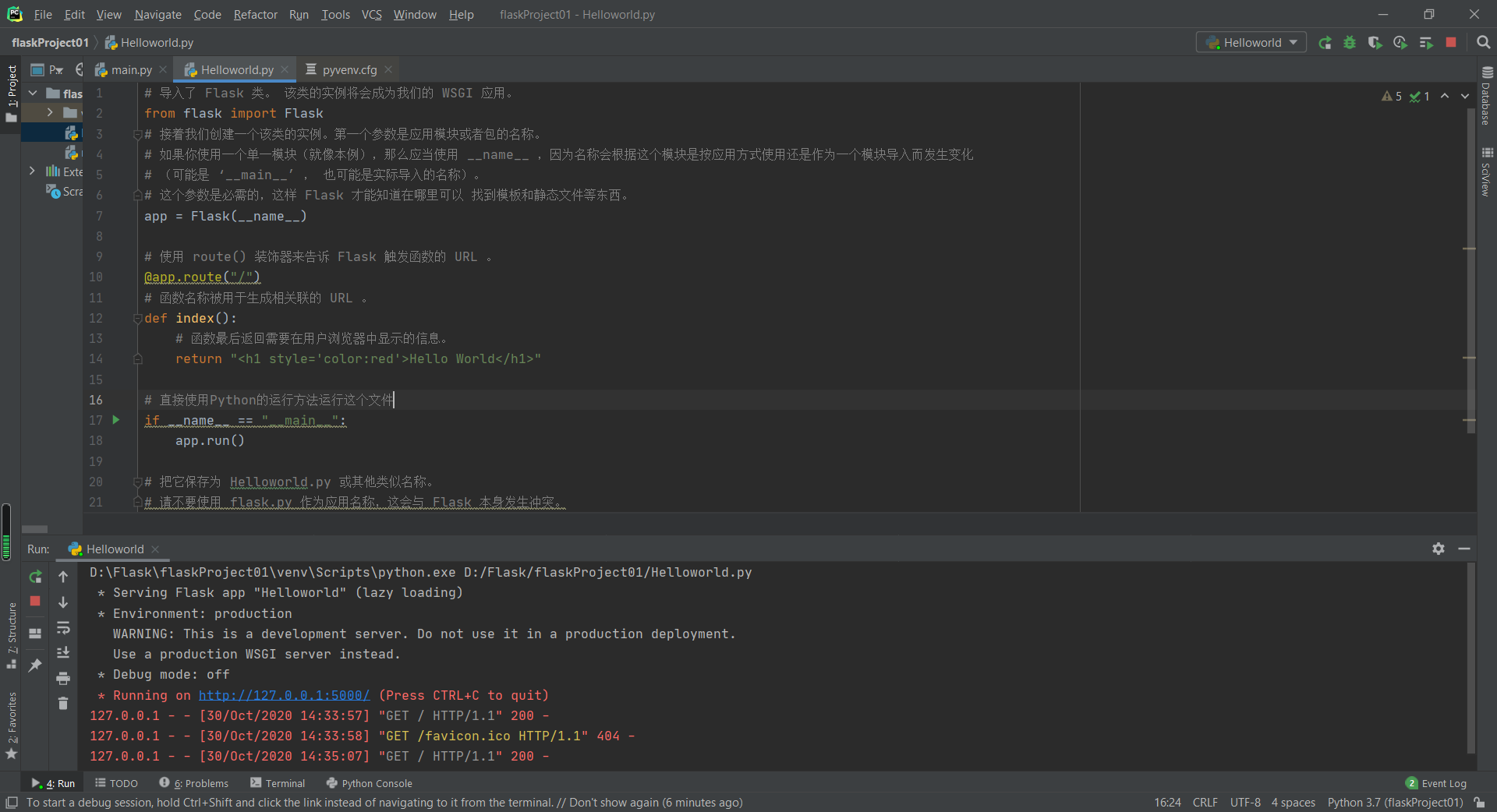


安装完成后可以看到页面下方出现package installed successfully的提示，这时我们就可以关闭这个页面了。

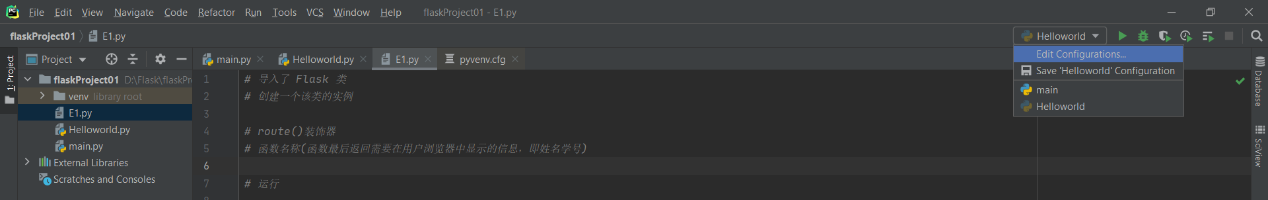
现在，我们这个项目的python解释器中看到和flask有关的内容。到这一步，整个python web项目的flask部分配置就已经完成啦。（当安装 Flask 时，以下配套软件会被自动安装。Werkzeug 用于实现 WSGI ，应用和服务之间的标准 Python 接口。Jinja 用于渲染页面的模板语言。MarkupSafe 与 Jinja 共用，在渲染页面时用于避免不可信的输入，防止注入攻击。ItsDangerous 保证数据完整性的安全标志数据，用于保护 Flask 的 session cookie.Click 是一个命令行应用的框架。用于提供 flask 命令，并允许添加自定义 管理命令。）



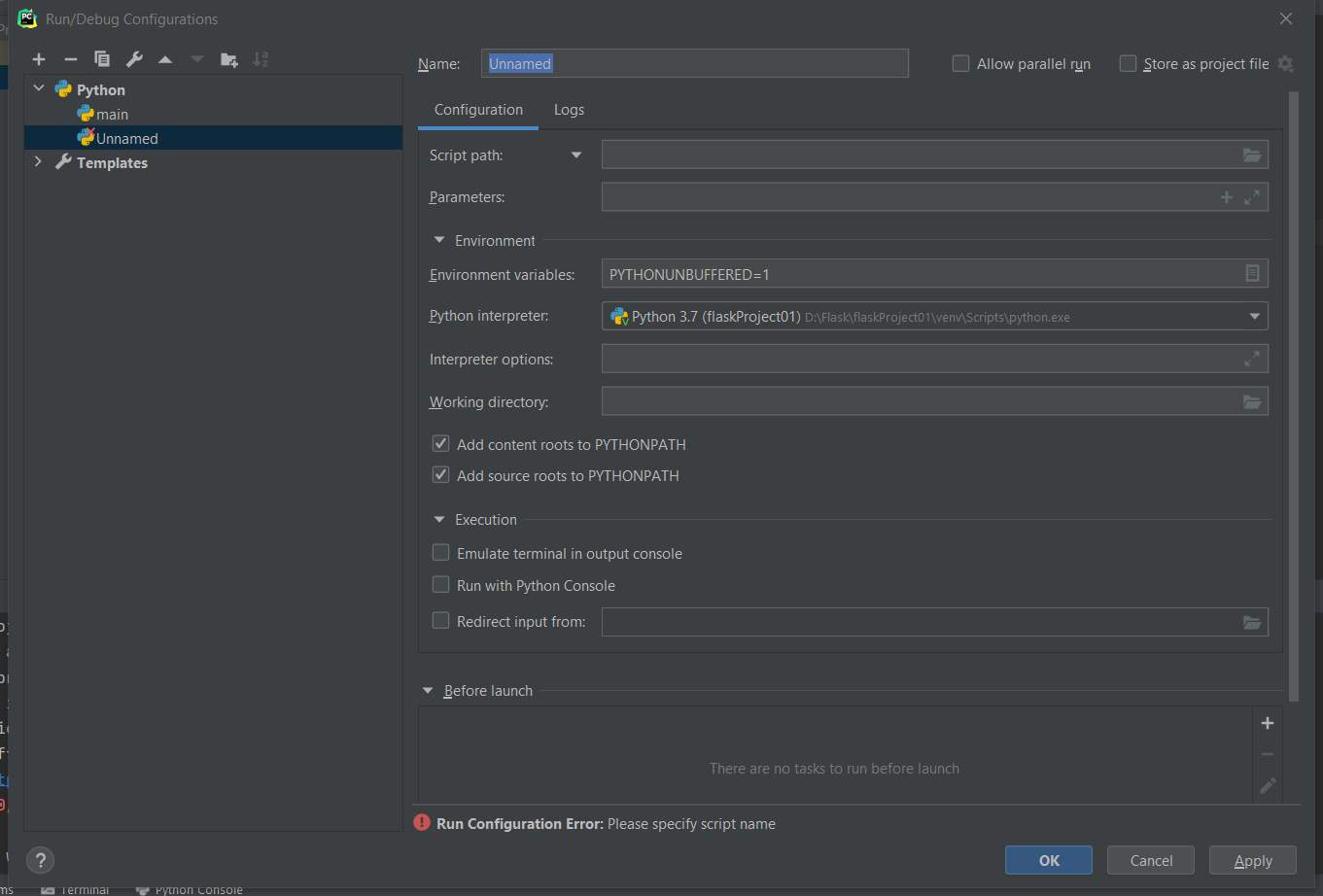
让我们试着用flask写一个HelloWorld文件。



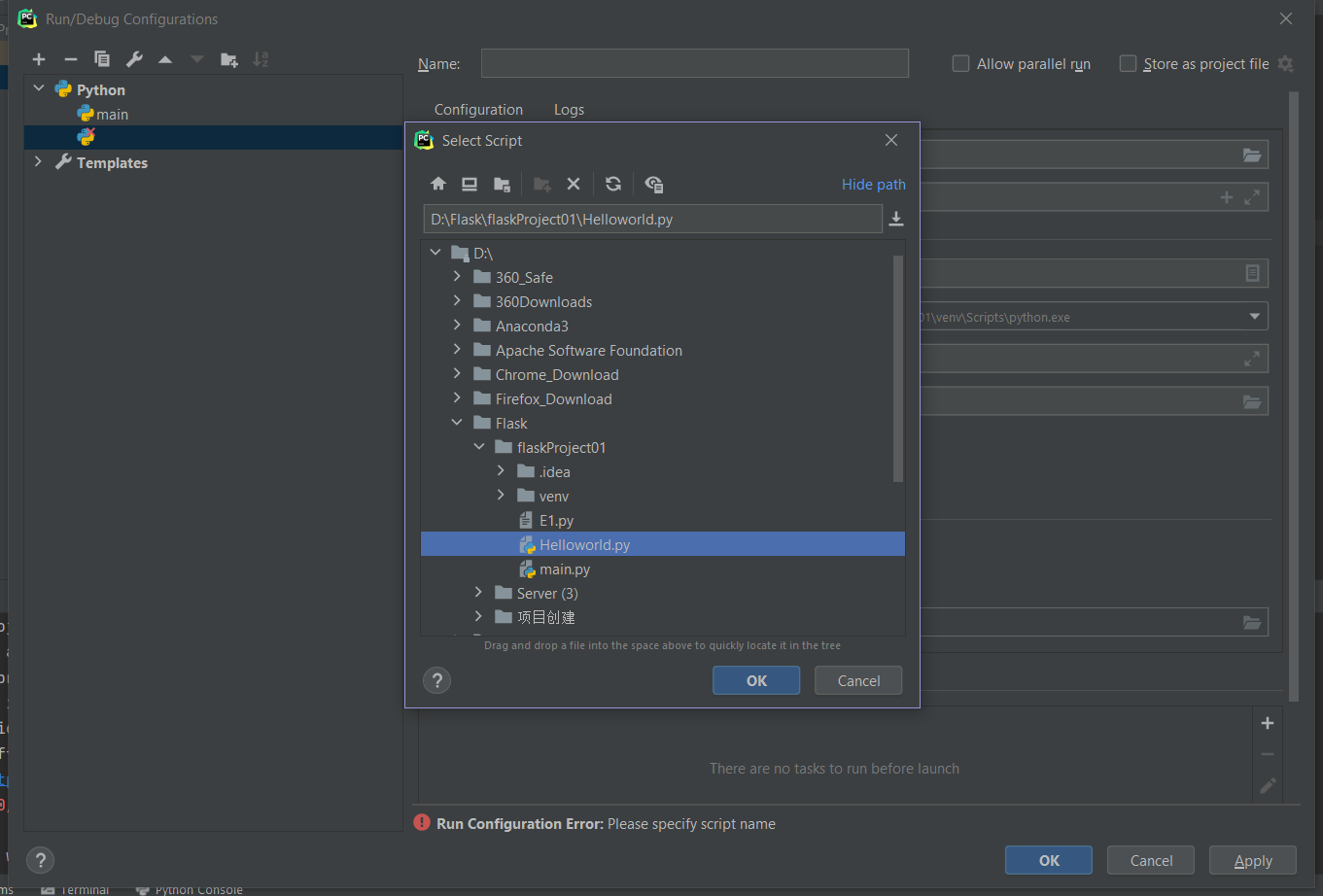
在运行前，我们需要先给这个文件创建运行环境。可以在PyCharm右上角的这个下拉菜单中的Edit Configurations中设置。



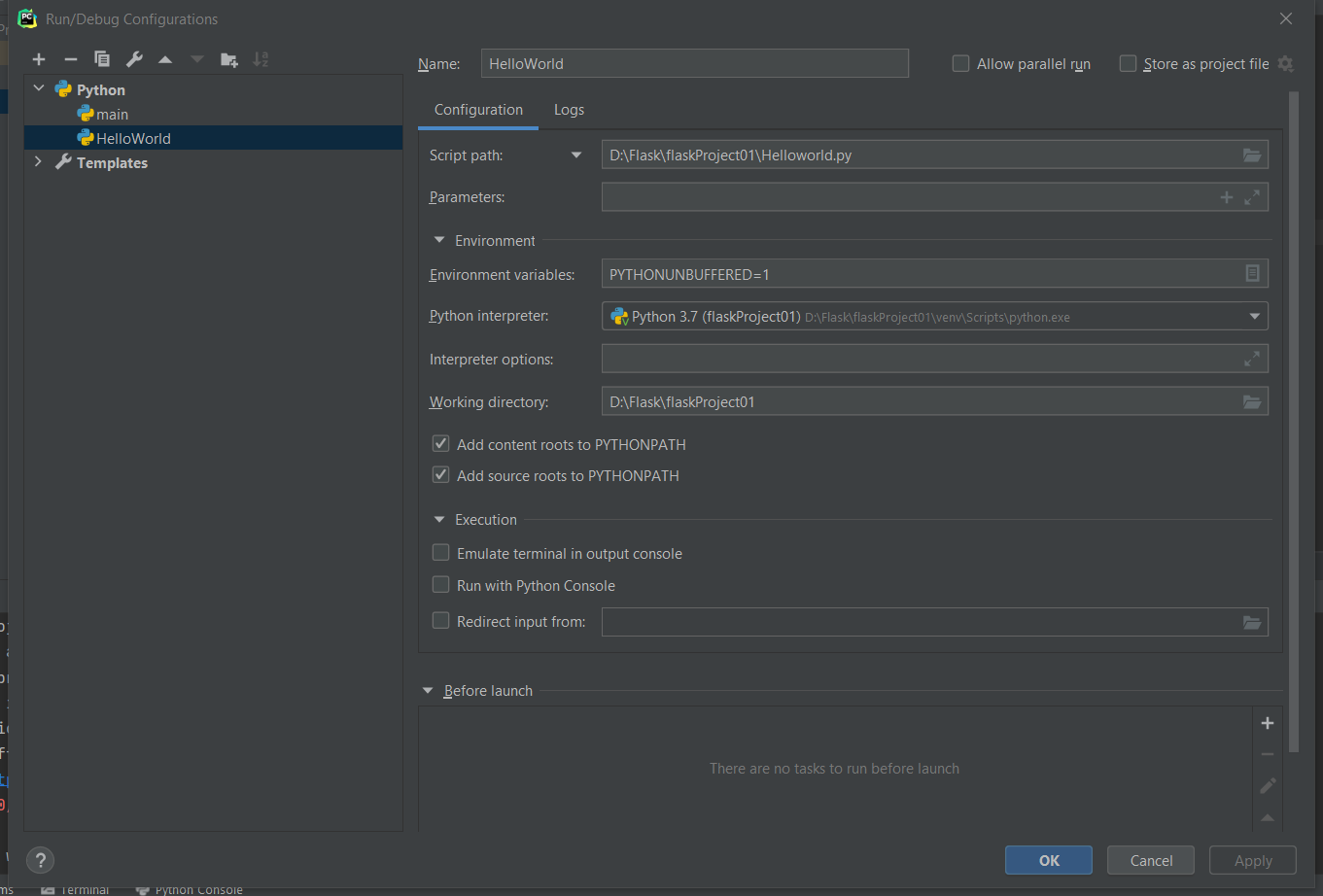
打开设置后选择左上角的＋添加一个Python可运行文件。



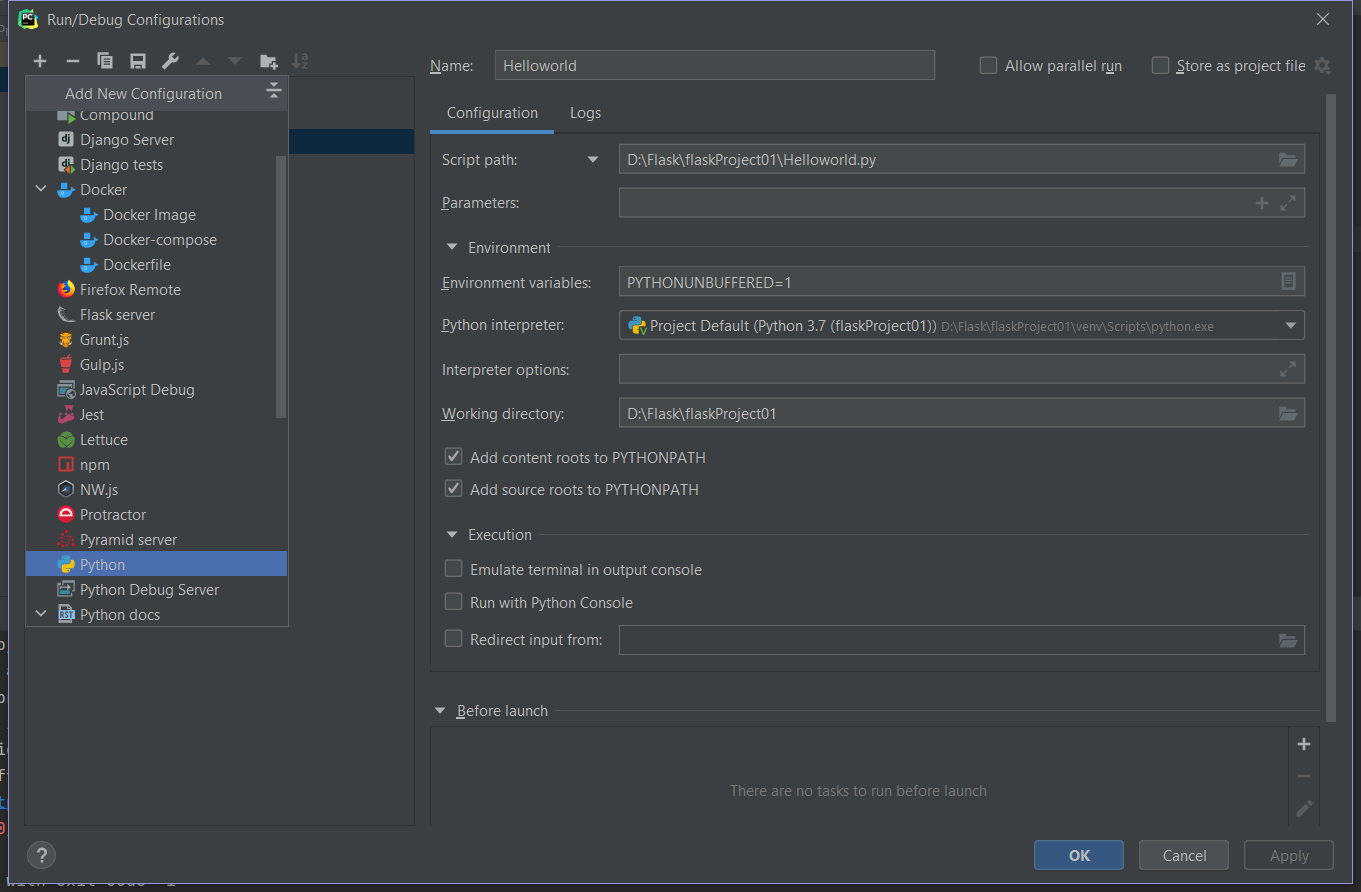
选择script，找到我们要添加的.py文件。



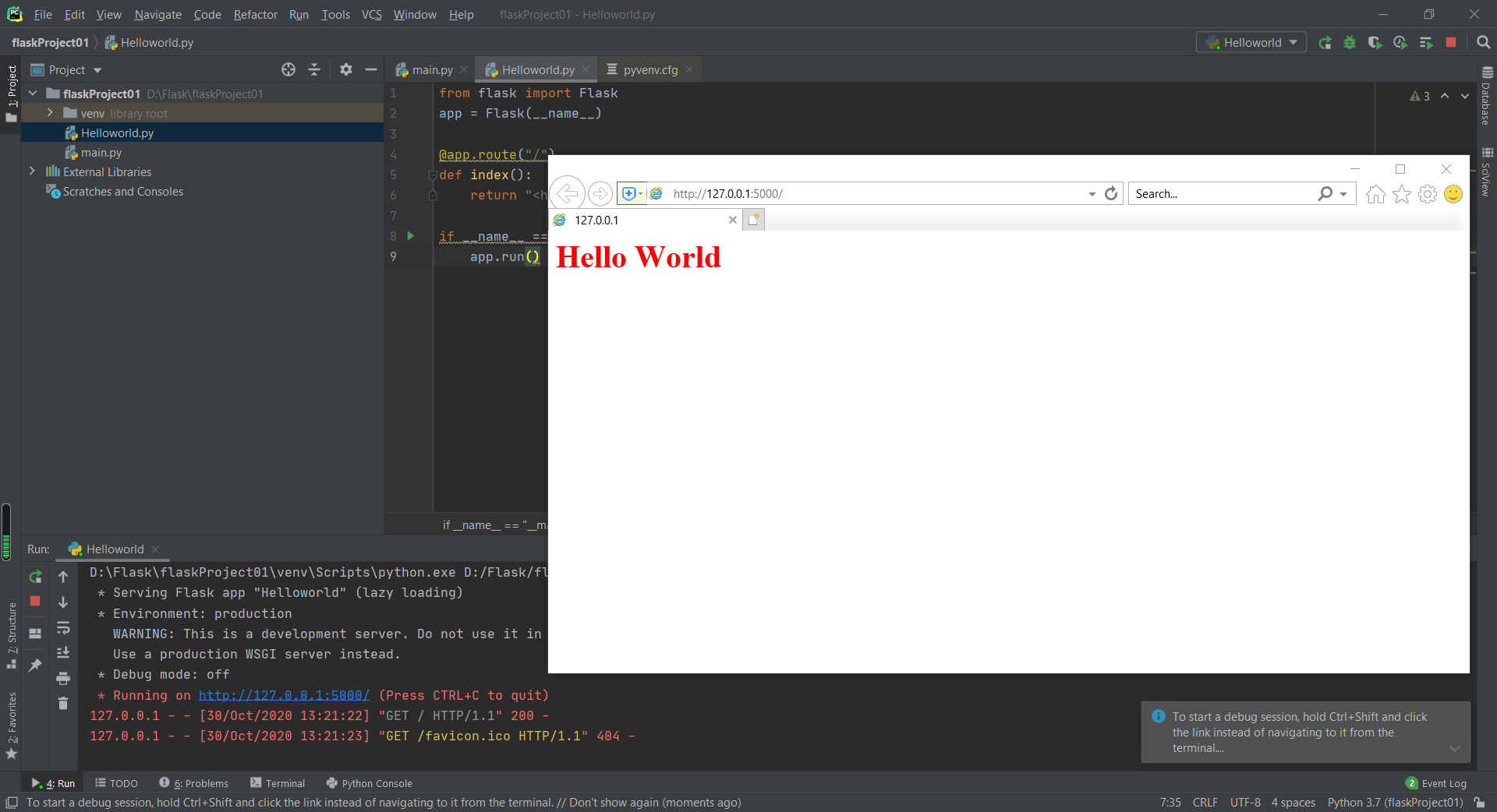
添加完成后点击确定即可。



添加完成后如下图所示。



这之后运行改文件就可以从http://127.0.0.1:5000/网页看到helloworld。



# 基于flask框架的小型博客系统

## 项目需求

接下来，我们将会创建一个名为myBlog的具备基本功能的博客应用。应用用户可以注册、登录和注销，用户还可以发布、编辑和删除自己帖子，同时，用户可以上传想要存储和分享的文件，也可以下载。

## 文件目录整理和分析

我们知道，一个简单的Flask应用可以使用单个.py文件实现，但当项目扩大时，好的目录结构和文件分割将会给代码带来更好的可读性。因此，在本项目中，我们也会使用包来管理代码，按照功能和性质将代码分成不同的模块，在需要的地方导入相应的代码模块实现对应功能。

项目的总目录中包含一个主文件包，我们可以将其命名为myBlog，一个测试模块，通常命名为tests，一个用于安装 Flask 和其他依赖的Python 虚拟环境包，通常命名为venv，项目的启动文件main.py和项目介绍README.md也存放在主文件包下。

本次项目的初始布局如下。

flaskProject01

|--myBlog/

| |--main.py

| |--\_\_init\_\_.py

| |--db.py

| |--schema.sql

| |--auth.py

| |--templates/

| | |--base.html

| | |--auth/

| | | |--login.html

| | | |--register.html

| |--static/

| |--style.css

|--tests/

|--venv/

## 定义和操作数据库

本次flask项目我们使用SQLite数据库来存储所需数据。Python内置了SQLite数据库支持，相应的模块为sqlite3。使用SQLite的便利性在于不需要单独配置一个数据库服务器，并且Python提供了内置支持。但是当并发请求同时要写入时，会比较慢一点，因为每个写操作是按顺序进行的。小应用没有问题，但是大应用可能就需要考虑换成别的数据库了，建议大家平时写项目就用mySQL或者mongoDB这类数据库，因为本次实践的重心不在数据库所以我们选择了这个很简便但功能不够好的数据库。

**连接数据库：**当使用SQLite数据库时，第一件事就是创建一个数据库的连接。所有操作都要通过该连接来执行。创建和连接数据库的代码写在myBlog/db.py文件中。

**创建表：**建表代码在myBlog/schema.sql中，本次课只需要一个用户表，表中包含用户的id用于识别该用户，同时存储了用户名和密码，要注意的是用户名是唯一的，并且用户名和密码都不能为空。

**DROP** **TABLE** **IF** **EXISTS** **user;**

**CREATE** **TABLE** **user** **(**

id INTEGER **PRIMARY** **KEY** AUTOINCREMENT**,**

username TEXT **UNIQUE** **NOT** **NULL,**

password TEXT **NOT** **NULL**

**);**

如果想要使用这个数据库，我们需要先初始化数据库文件。初始化数据库的指令如下：

**set FLASK\_APP=myBlog**

**set FLASK\_ENV=development**

**flask init-db**

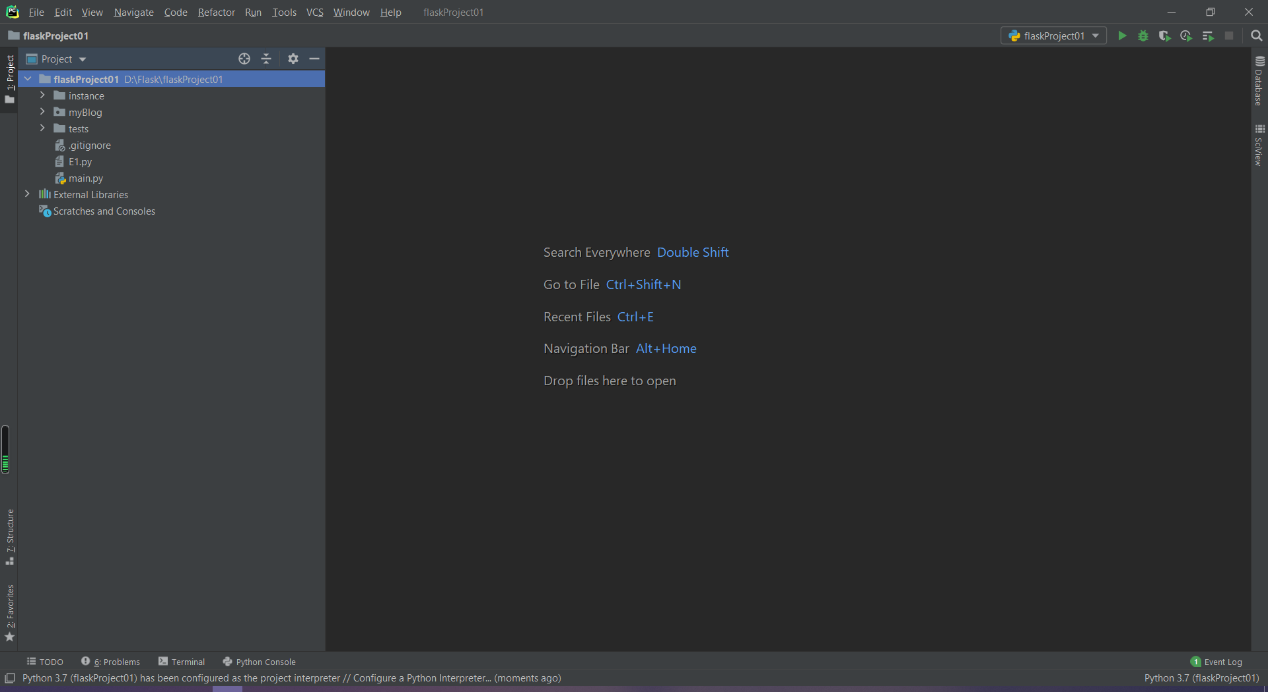
这段初始化我们直接在terminal输入，PyCharm的terminal在整个页面最下面一行中可以找到。

初始化数据库文件后会在instance 文件夹中生成一个myBlog.sqlite文件，这里我们已经生成好了这个文件，同学们可以直接使用。

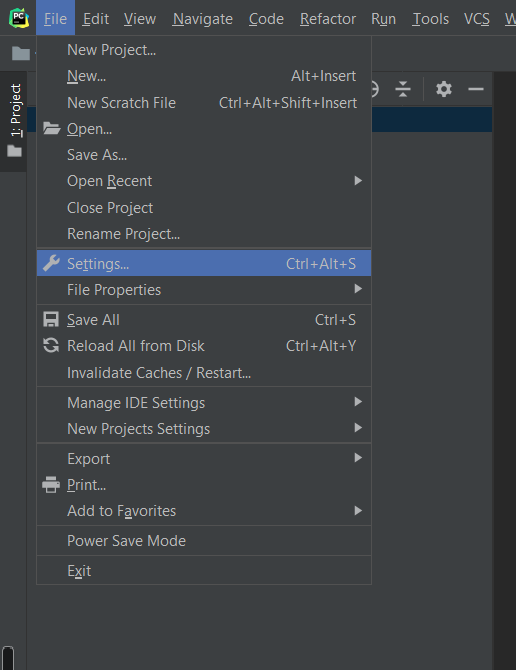
## 使用PyCharm运行项目

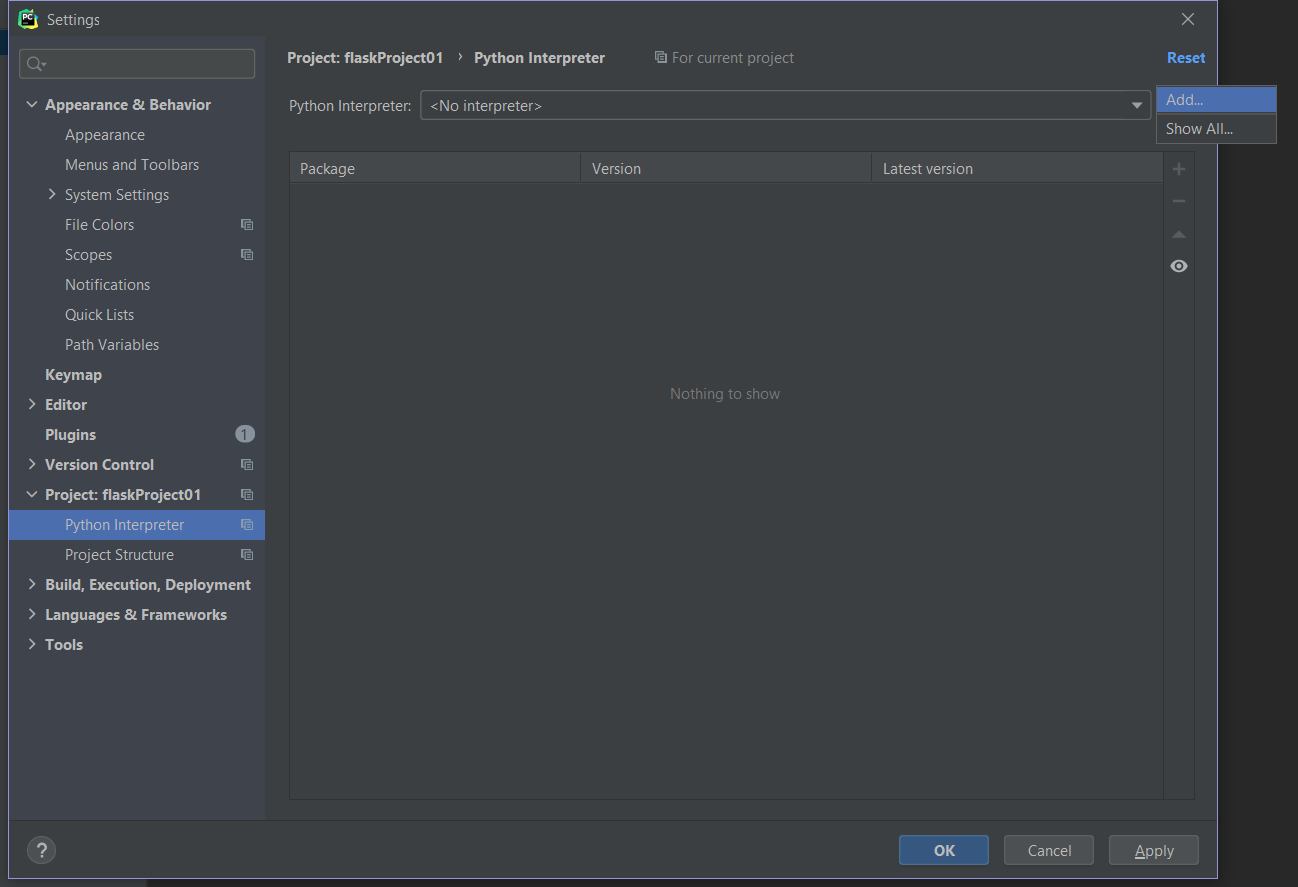
首先打开项目，项目地址为：

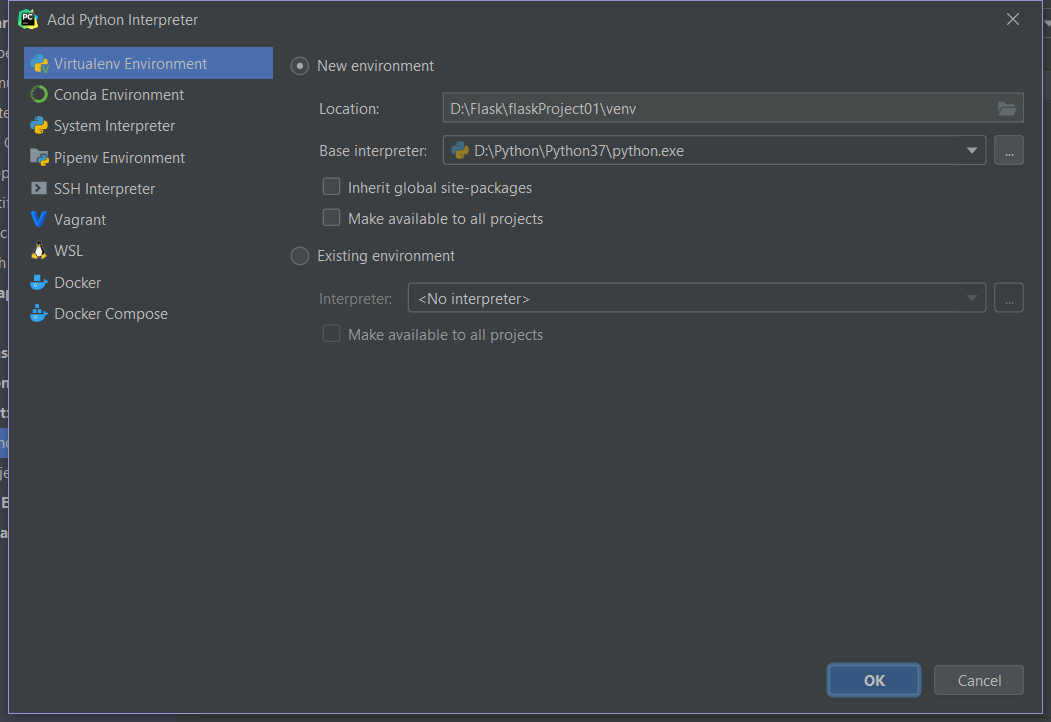
https://bhpan.buaa.edu.cn:443/link/768EE3D0CE34222FA31233EF3608AB0C

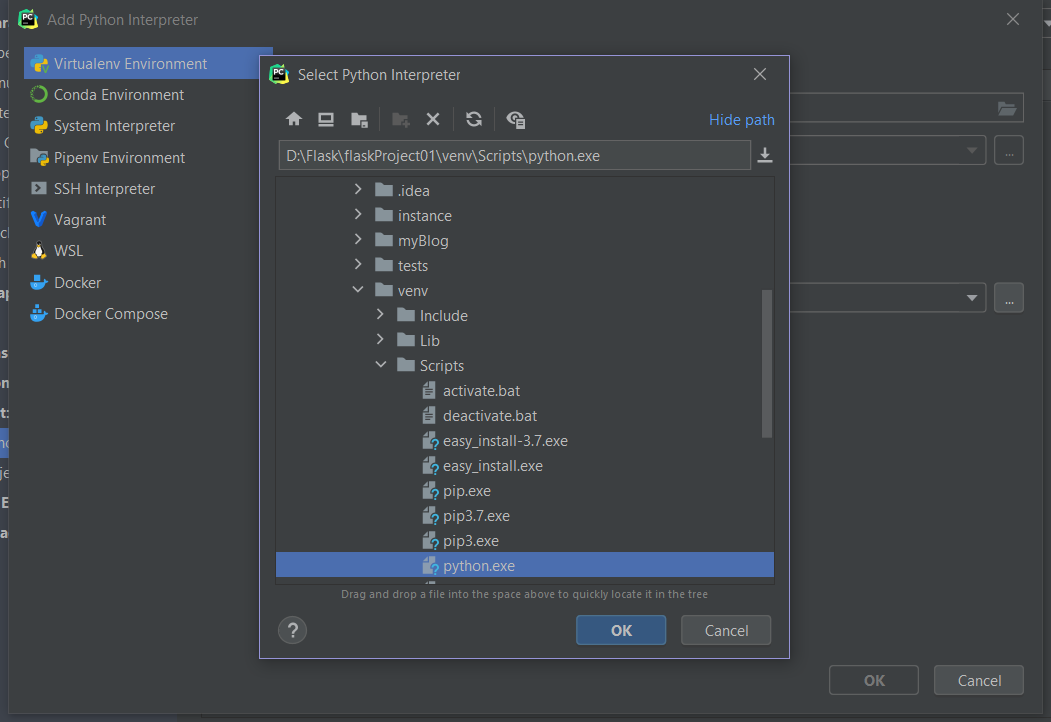


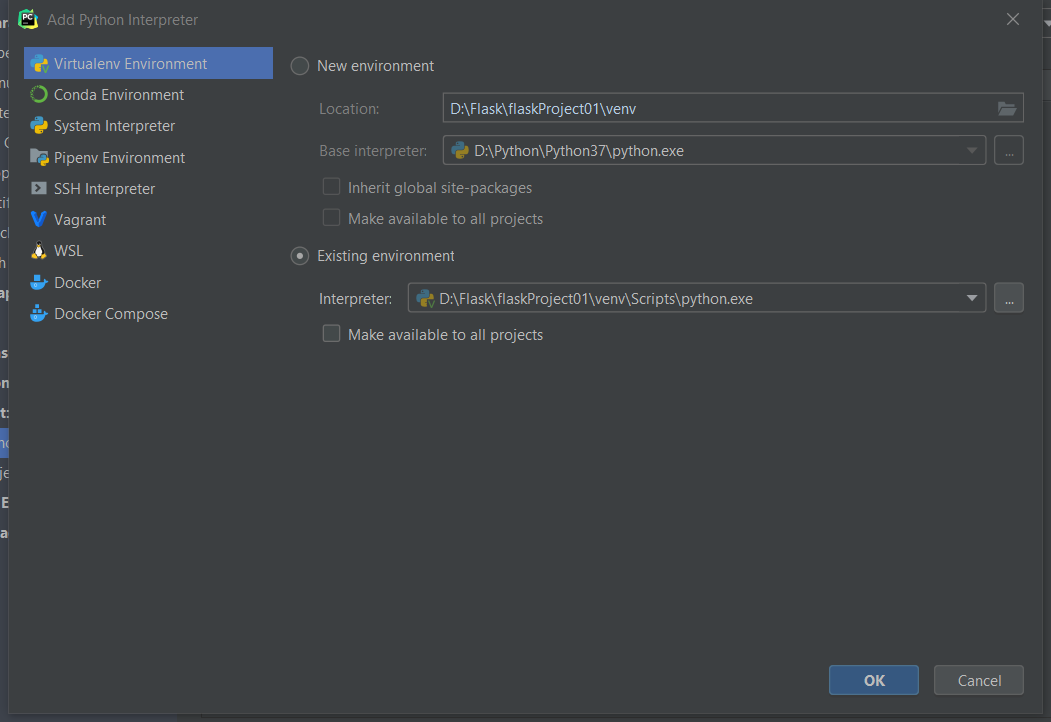
选择设置部分添加该项目的虚拟环境。



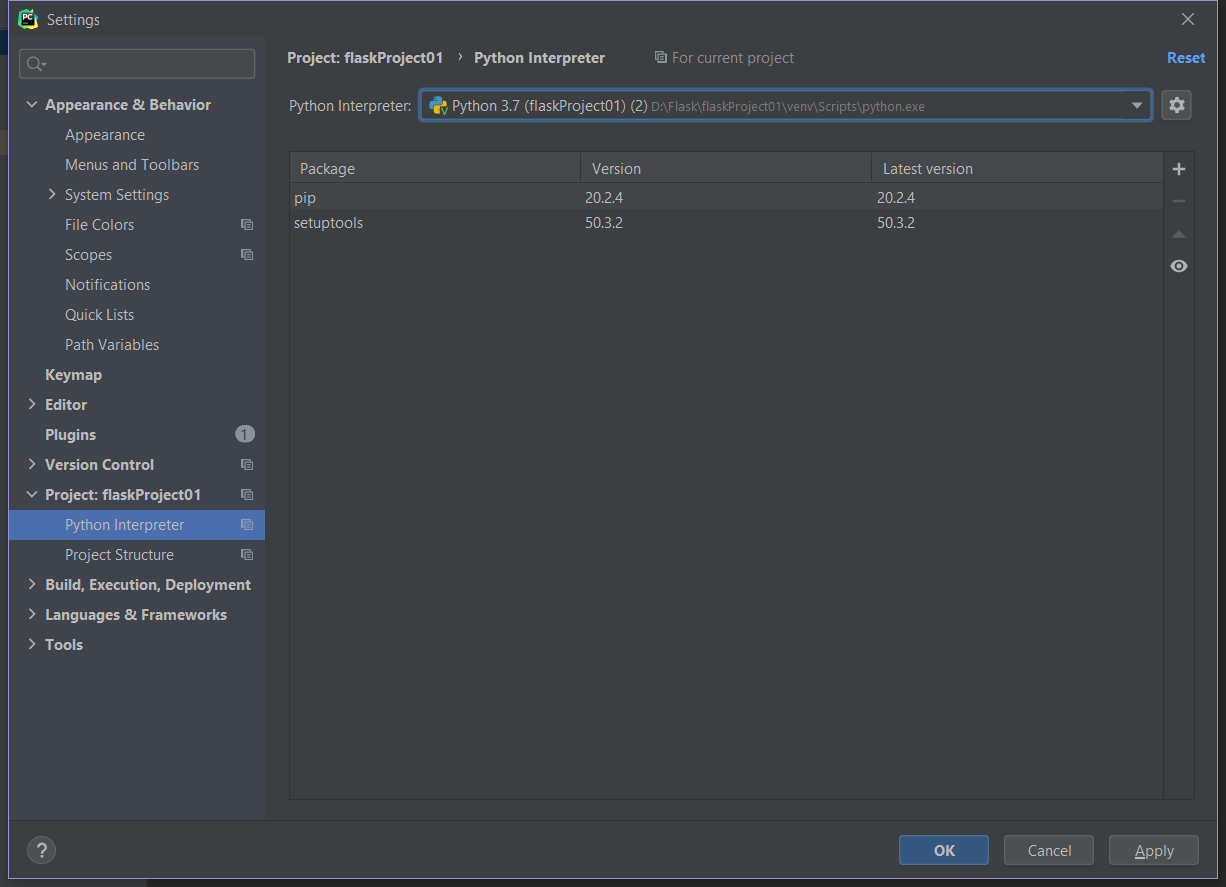




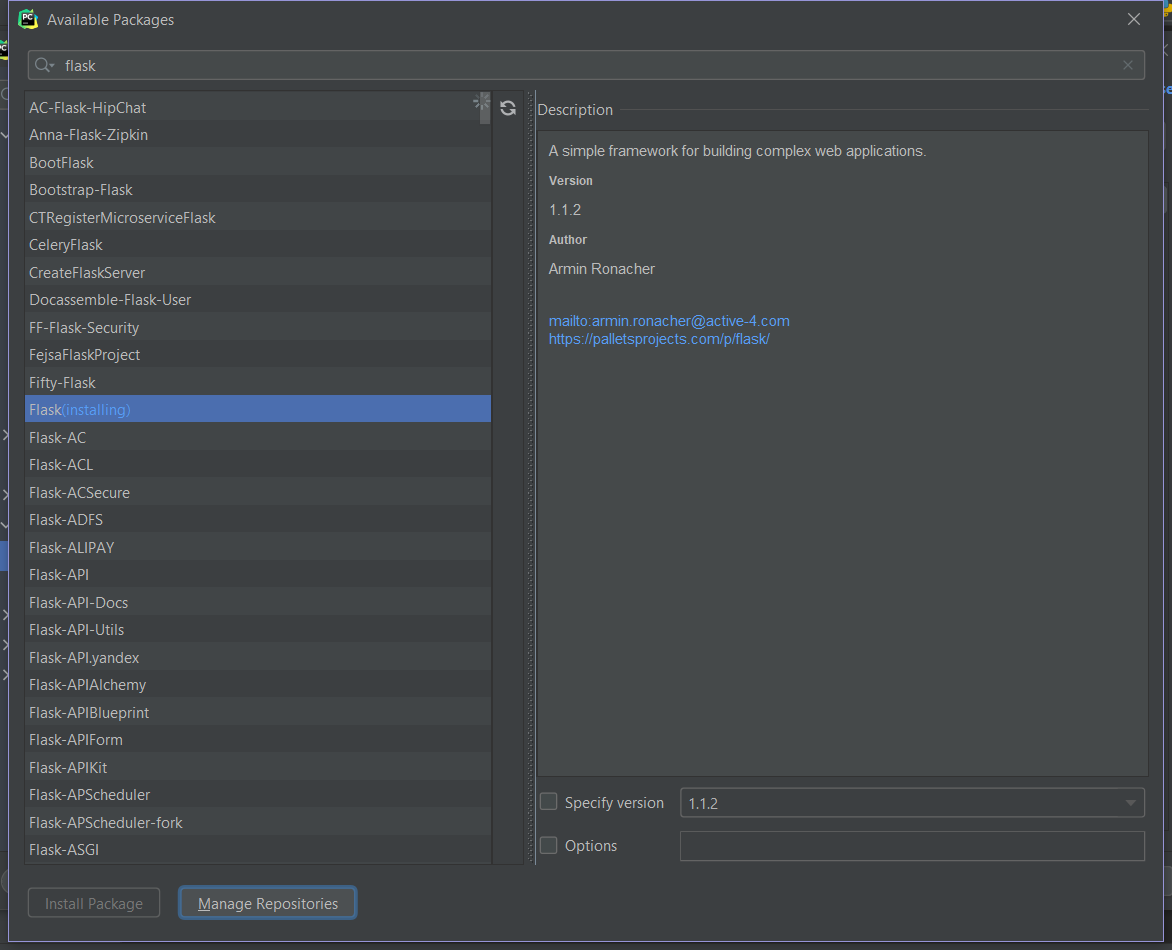


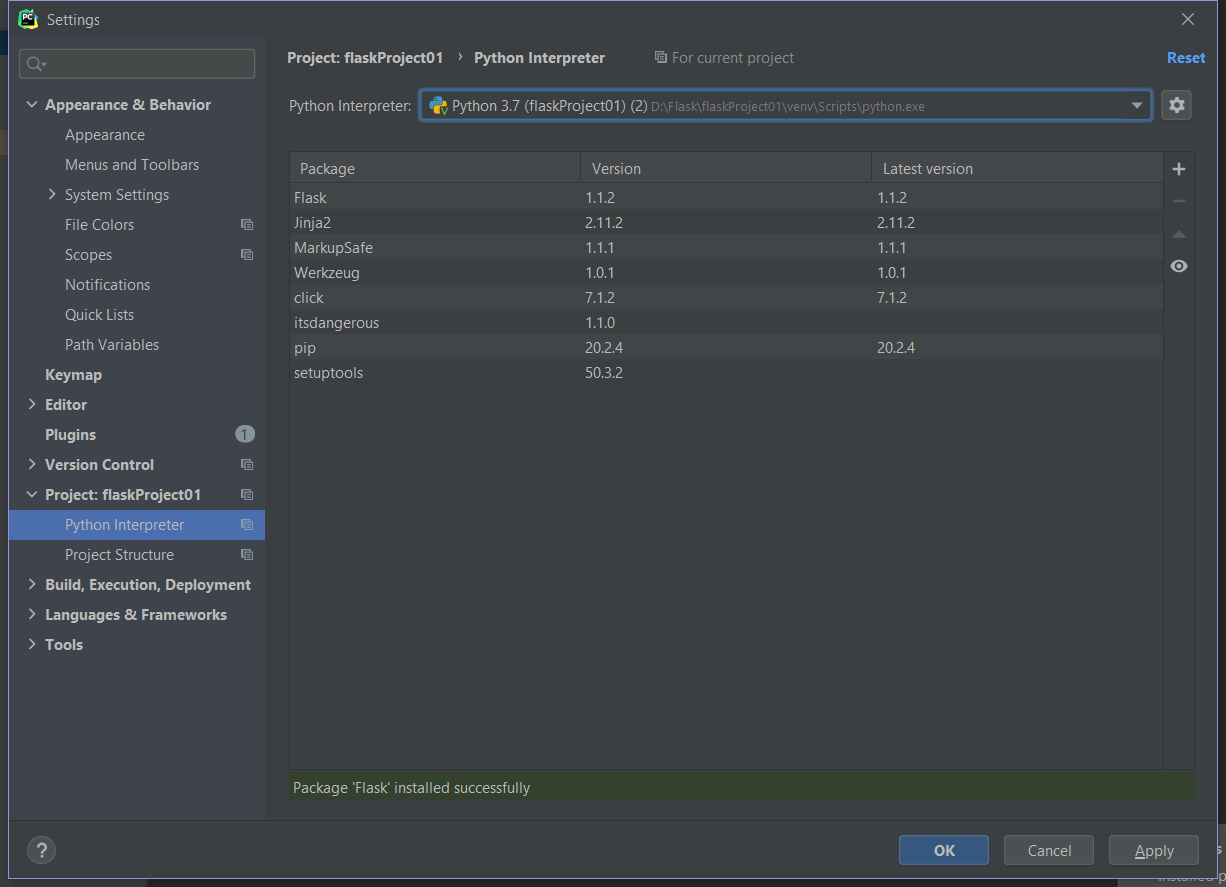


Python环境设置好后显示如下。



现在我们需要添加flask环境





**先初始化数据库：**

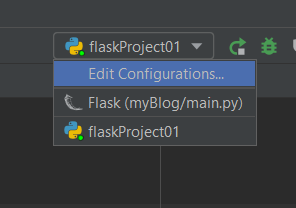
**set FLASK\_APP=myBlog**

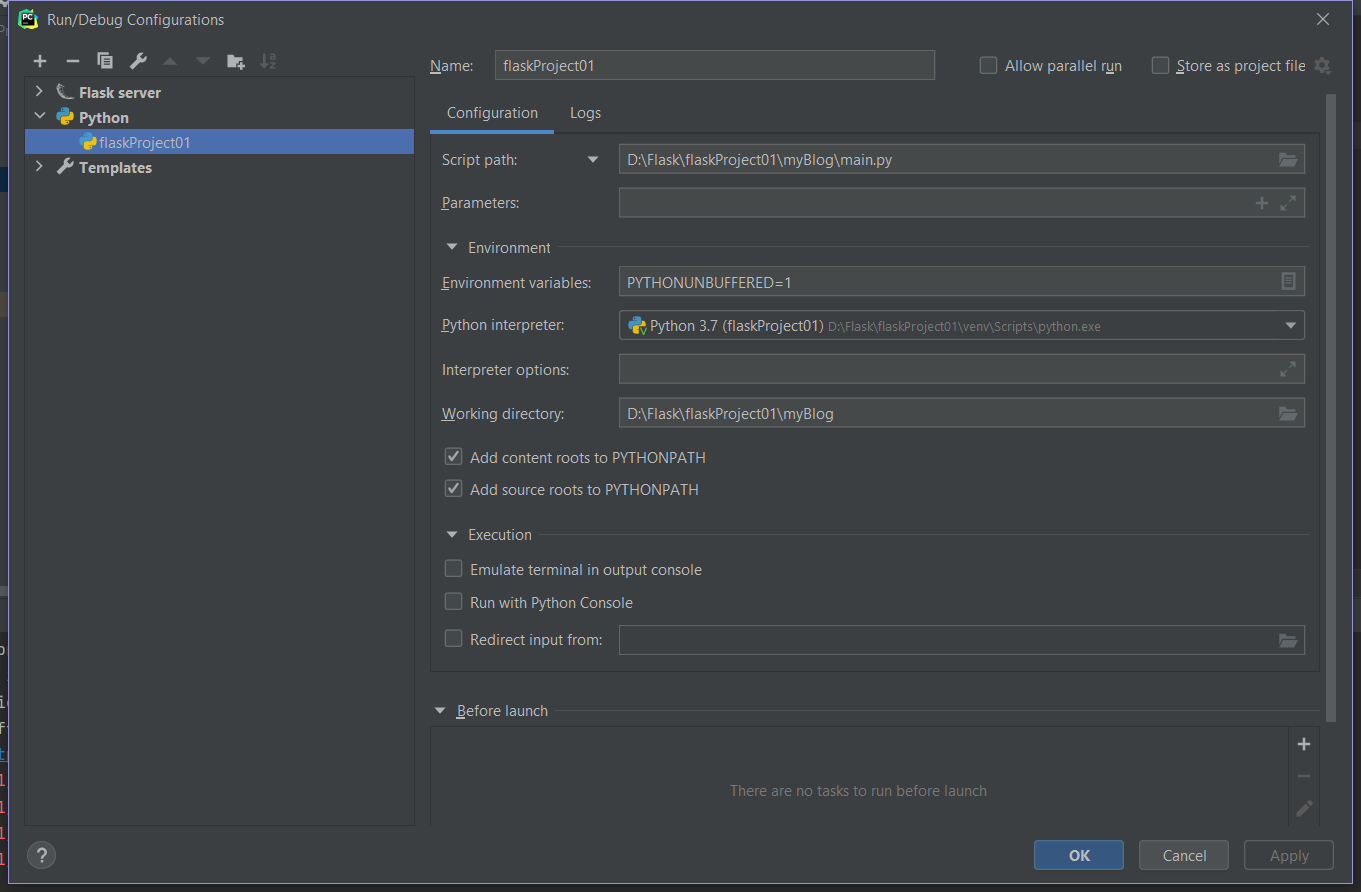
**set FLASK\_ENV=development**

**flask init-db**

**也可以直接在terminal里面输入flask run**

**接下来即可运行项目的可运行文件main文件。**





设置完成后可以在网页中看到相应界面。

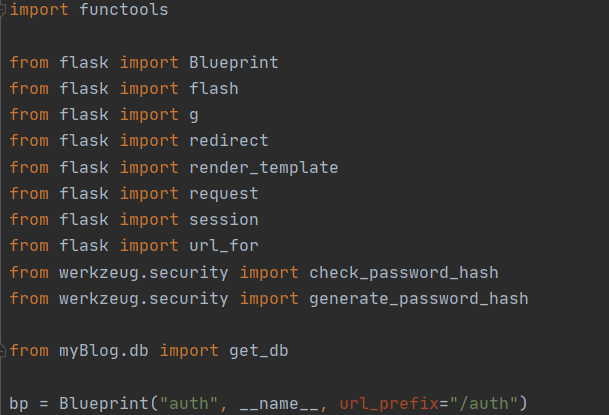
## 编写注册登录功能

### 蓝图编写

蓝图是一种组织一组相关视图及其他代码的方式。与把视图及其他代码直接注册到应用的方式不同，蓝图方式是把它们注册到蓝图，然后在工厂函数中把蓝图注册到应用。

使用博客首先需要认证，因此我们先写认证蓝图。

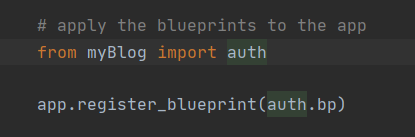
myBlog/auth.py



这里创建了一个名称为 'auth' 的 Blueprint。蓝图需要知道是在哪里定义的，因此把 \_\_name\_\_ 作为函数的第二个参数。 url\_prefix 会添加到所有与该蓝图关联的URL前面。

接下来，我们在初始化文件中编写app.register\_blueprint方法导入用户认证蓝图。新的代码放在工厂函数的尾部返回应用之前。

myBlog/\_\_init\_\_.py



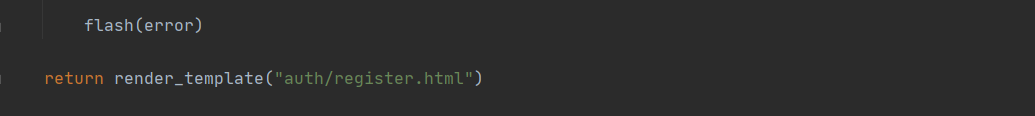
认证部分包括三个内容，分别是注册、登录和注销。这里我们重点讲解注册登录，请大家课后自行完成注销部分。

### 注册功能

当用访问 /auth/register URL 时，注册方法会返回用于填写注册内容的表单的前端界面。当用户提交表单时，系统会验证表单内容，然后要么再次显示表单并显示一个出错信息，要么创建新用户并显示登录页面。

myBlog/auth.py

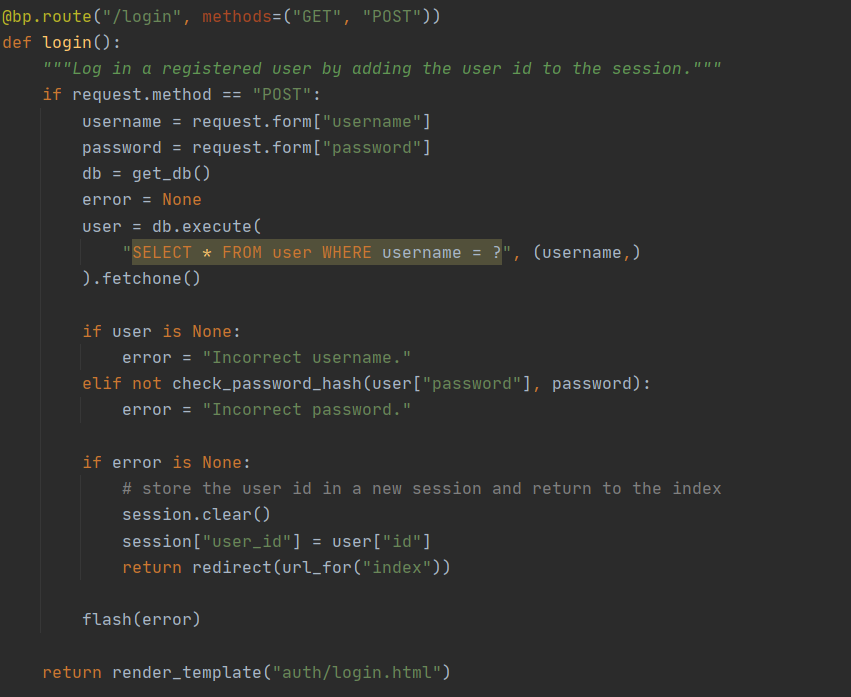




### 登录功能

登录部分的函数如下。

myBlog/auth.py



### 注销功能

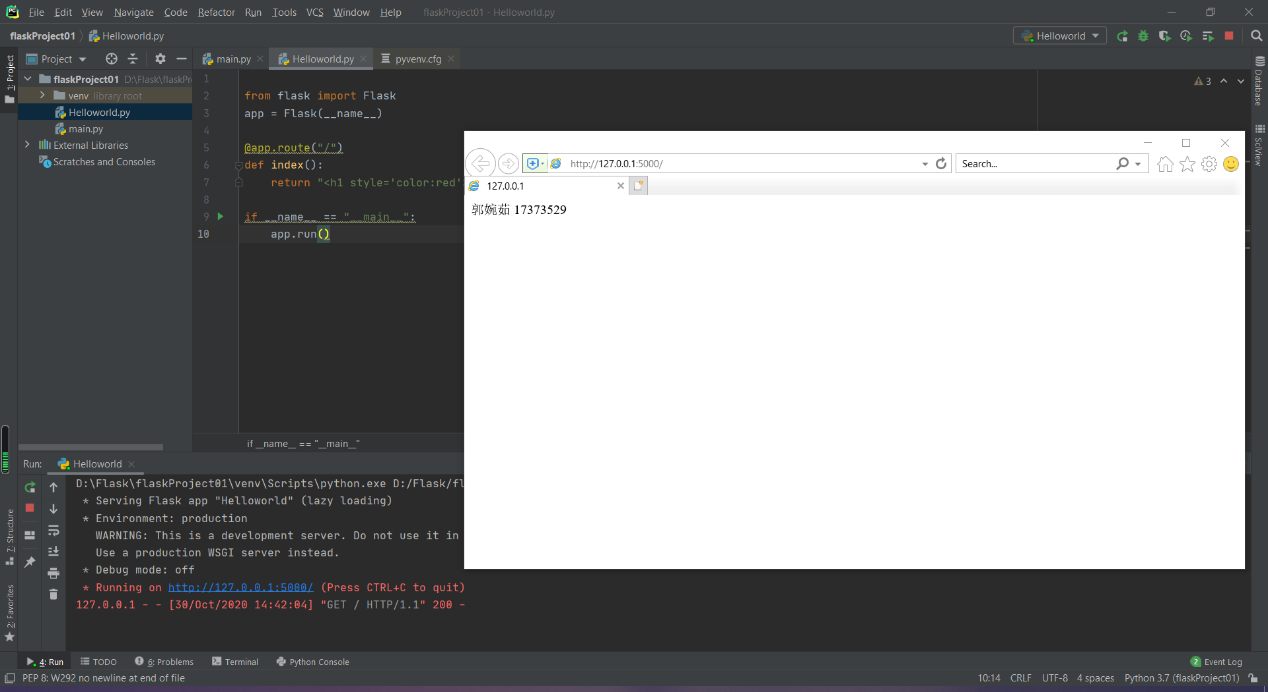
注销的时候需要把用户 id 从session中移除。 然后load\_logged\_in\_user就不会在后继请求中载入用户了。大家添加一个函数就可以完成用户的注销。

# 练习

## 签到（1分）

## Flask项目的运行（2分）

使用flask在页面中显示姓名学号字段，请截图完整的网址和页面，示例如下。



## 注销功能实现（2分）

按照课上所讲在myBlog/auth.py文件下的logout()函数中完成注销功能。请在完成后演示整个注册登录和注销流程。