Flask整合ECharts

前言

ECharts作为一种商业图表分析工具，在进行简单数据可视化时，我们会将数据直接写入HTML代码中，但是，随着数据量的不断增多，如果按照之前直接写入HTML文件的方式继续的话，代码会变得不利于维护。因此，在以后的ECharts可视化开发中，提倡前后端分离开发，这里的后端主要功能是提取可视化数据、进行数值计算并将计算结果整合到Echarts图表中。

主要技术栈

- Flask 轻量级WEB开发框架
- jinja2 模板渲染工具
- ECharts 商业图表分析工具

核心技术详解

Flask

Flask简介

Flask是一个基于Python开发的轻量级WEB开发框架，通过非常简洁的语法就可以搭建自己的Web服务器，下面通过一个简单的示例来认识Flask程序的基本框架。

入门案例

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'hello world!'

if __name__ == "__main__":
    app.run()
```

代码解析

代码	释义
from flask import Flask	从flask库中导入Flask类
app = Flask(**name**)	使用**name**实例化Flask对象
@app.route('/')	url装饰器:给hello()方法配置url使得hello()可以在浏览器中被访问
def hello(): return 'hello world!'	定义hello()方法，返回'hello world!'字符串
if **name** == "**main**":	程序执行的入口
app.run()	启动服务

代码效果演示

代码执行流程分析

首先，找到程序入口，顺序执行入口下方的代码段(这里只有一行app.run()),开启服务，此时后端服务已经启动，根据程序执行结果提示，我们知道我们的服务开启于本地的5000端口，使用浏览器访问本地的5000端口。然后根据app.route('/')找到对应的方法(hello()视图函数)，将视图函数返回的内容(可以是字符串也可以是HTML格式的字符串也可以是html文件)响应给浏览器，浏览器会自动解析内容，并最终显示在屏幕上。

深入了解Flask

url中的变量

在app.route()中定义的url规则，是针对该装饰器下方的函数生效的，目的在于使得该函数可以在网页中被浏览到。同时在定义的url规则中，可以设定响应的变量，格式为“<变量名>”，url装饰器会解析url中的对应的值并赋值给对应的变量，并且这个变量可以在程序中被使用。

```
from markupsafe import escape

@app.route('/user/<name>')
def user_page(name):
    return f'User: {escape(name)}'
```

注意：这里escape是对变量的转义处理，目的在于防止用户名称中包含恶意代码

url中变量的类型转换

上述所说的变量不止字符串类型，如果现在想要提取url中的数字作为变量并且要类型统一呢？就需要用到Flask的变量类型转换器了。Flask为我们提供了七种类型转换器：

```
#: the default converter mapping for the map.
DEFAULT_CONVERTERS = {
    'default':          UnicodeConverter,
    'string':           UnicodeConverter,
    'any':              AnyConverter,
    'path':             PathConverter,
    'int':              IntegerConverter,
    'float':            FloatConverter,
    'uuid':             UUIDConverter,
}
```

类型转换器的使用代码如下：

```
@app.route('/user/<int:userid>')
def user(userid):
    print(userid)
    print(type(userid))
    # 因为是int类型，所以返回时要先转换成string
    return str(userid)
```

上述代码运行后，若我们访问127.0.0.1:5000/user/14，装饰器会解析我们访问的url得到变量userid的值。其实不难发现我们的userid就是14。

多url映射同一视图函数

使用多个路由映射同一个视图函数，简单来说就是访问多个url都是同一个资源。

```
@app.route('/')
@app.route('/index')
@app.route('/home')
def hello():
    return 'Welcome to My Watchlist!'
```

上面这个例子，不论访问127.0.0.1：5000/还是访问127.0.0.1：5000/index还是访问127.0.0.1：5000/home都是同一个网页。

jinja2

jinja2是一个基于Python的模板渲染工具。简单来说，它的功能就是渲染数据。

基本概念

模板：HTML格式

渲染：将数据填充到HTML中

基本语法

```
{{ 变量名 }}
{% python语句 %}
{# 注释 #}
```

jinja2变量过滤器

变量过滤器是对接收进来的变量进行相关的转换操作比如，大小写字母转换，数据类型转换，字符串首尾去空格等。其基本语法如下：

```
{{ 变量名 | 过滤器名称 }}
```

jinja2常用的过滤器

过滤器名称	说明
safe	渲染时值不转义
capitialize	把值的首字母转换成大写，其他子母转换为小写
lower	把值转换成小写形式
upper	把值转换成大写形式
title	把值中每个单词的首字母都转换成大写
trim	把值的首尾空格去掉
striptags	渲染之前把值中所有的HTML标签都删掉
join	拼接多个值为字符串
replace	替换字符串的值
round	默认对数字进行四舍五入，也可以用参数进行控制
int	把值转换成整型

提示：变量过滤器支持多重过滤，语法如下：

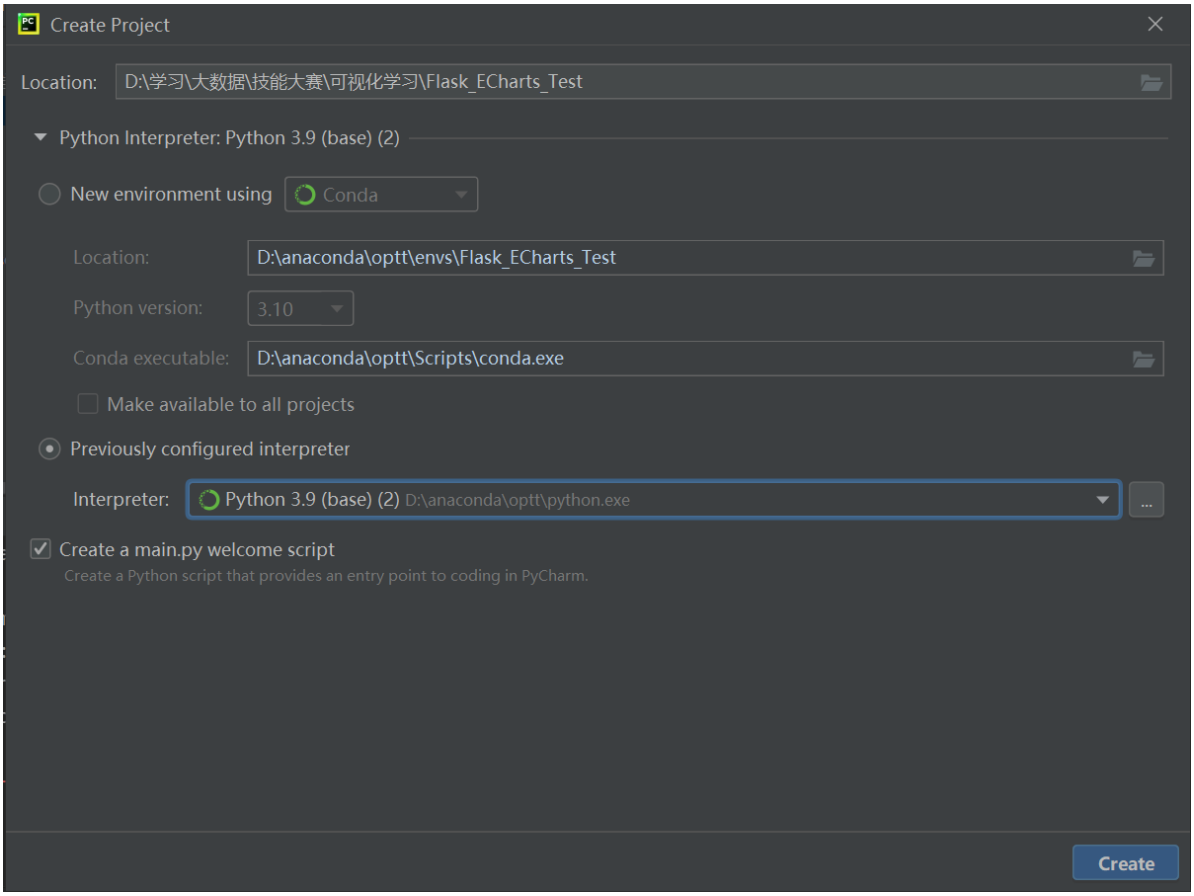
```
{{ 变量名 | 过滤器1 | 过滤器2 }}
```

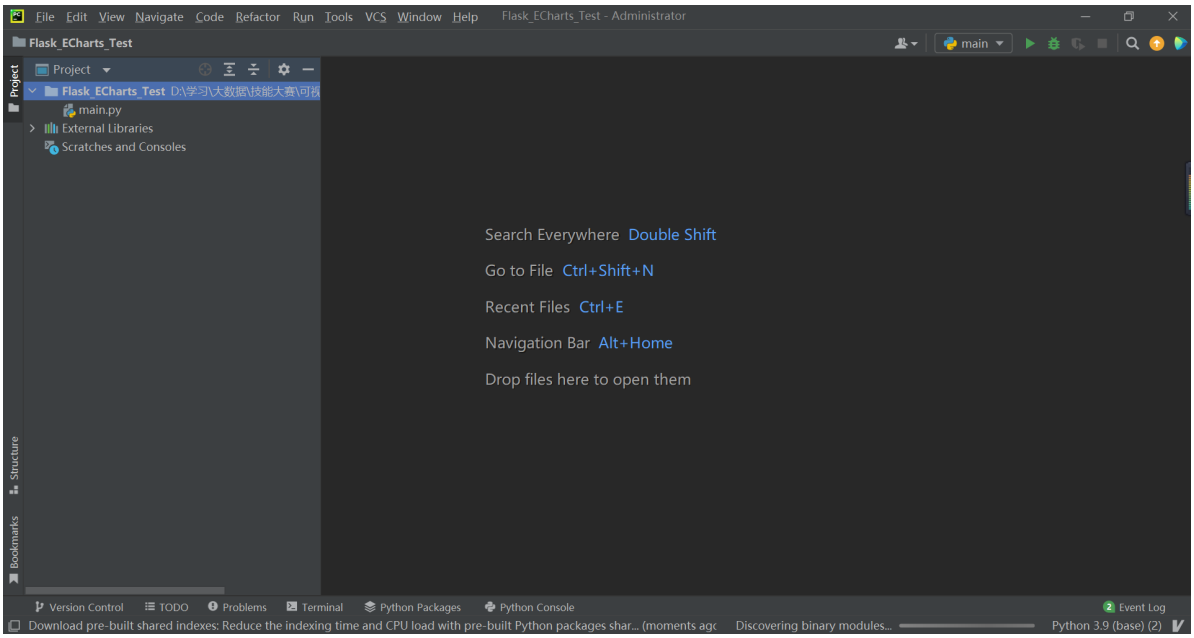
ECharts

入门教程详见：[echarts学习笔记(一) - shineQ - 博客园 (cnblogs.com)]

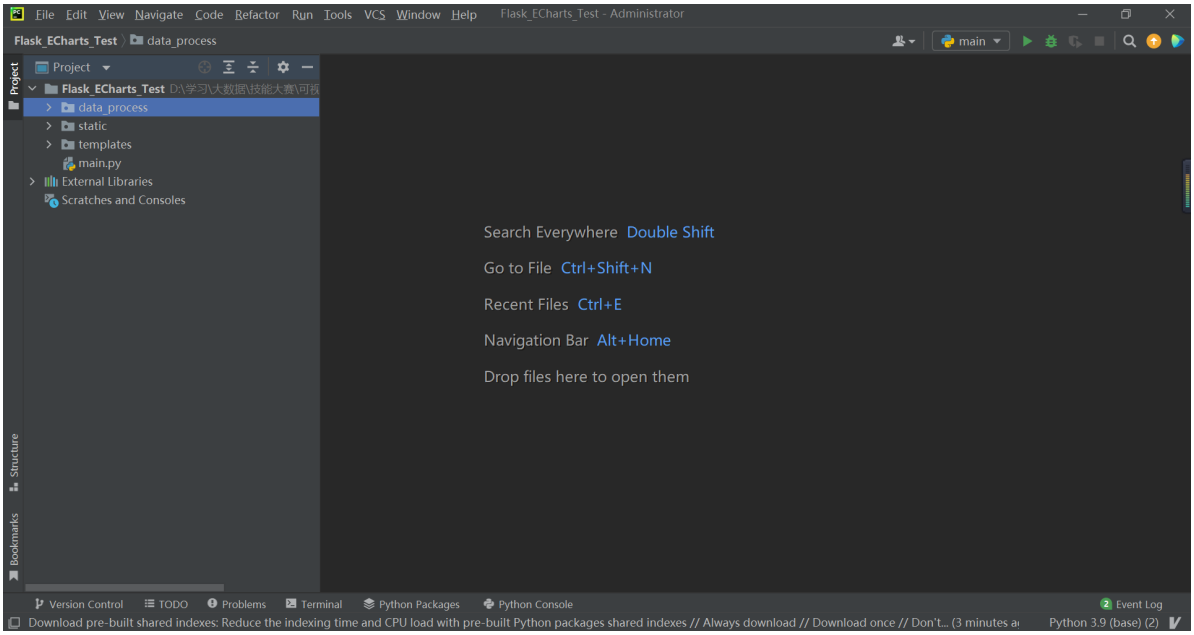
Flask+ECharts整合实战

创建项目


Create Project
Location:
D:\学习\大数据\技能大赛\可视化学习\Flask_ECharts_Test
Python Interpreter: Python 3.9 (base) (2)
New environment using
Conda
Location:
D:\anaconda\optt\envs\Flask_ECharts_Test
Python version:
3.10
Conda executable:
D:\anaconda\optt\Scripts\conda.exe
Make available to all projects
Previously configured interpreter
Interpreter:
Python 3.9 (base) (2) D:\anaconda\optt\python.exe
Create a main.py welcome script
Create a Python script that provides an entry point to coding in PyCharm.
Create


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Flask_ECharts_Test - Administrator
Flask_ECharts_Test
main
Project
Flask_ECharts_Test D:\学习\大数据\技能大赛\可视
main.py
External Libraries
Scratches and Consoles
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open them
Version Control
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shar... (moments ago
Discovering binary modules...
Python 3.9 (base) (2)

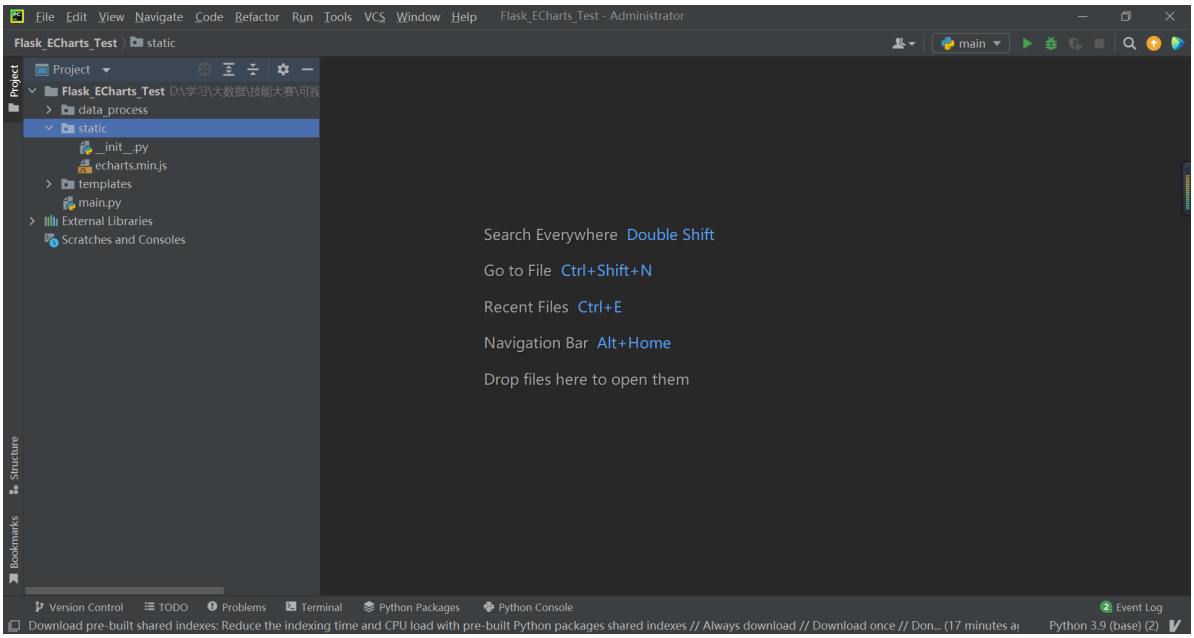
创建项目文件存储目录



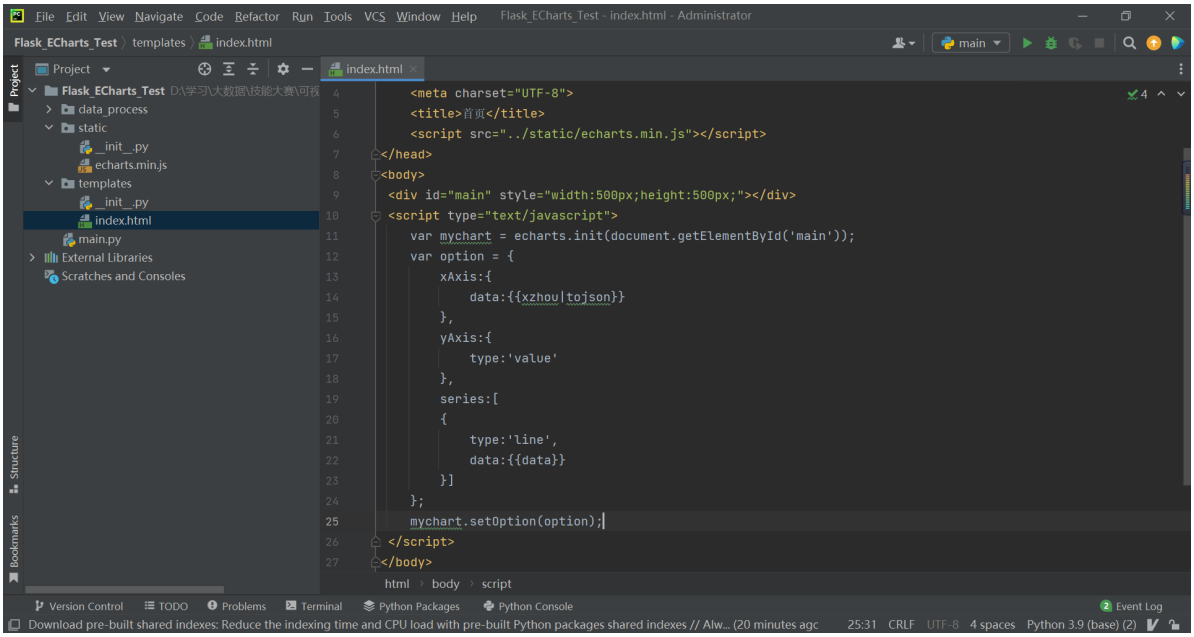
存储目录解析

- templates主要存放编写的HTML文件
- static主要存放编写HTML文件所需要的js脚本以及可视化所需要的数据文件
- data_process主要存放处理数据的python程序

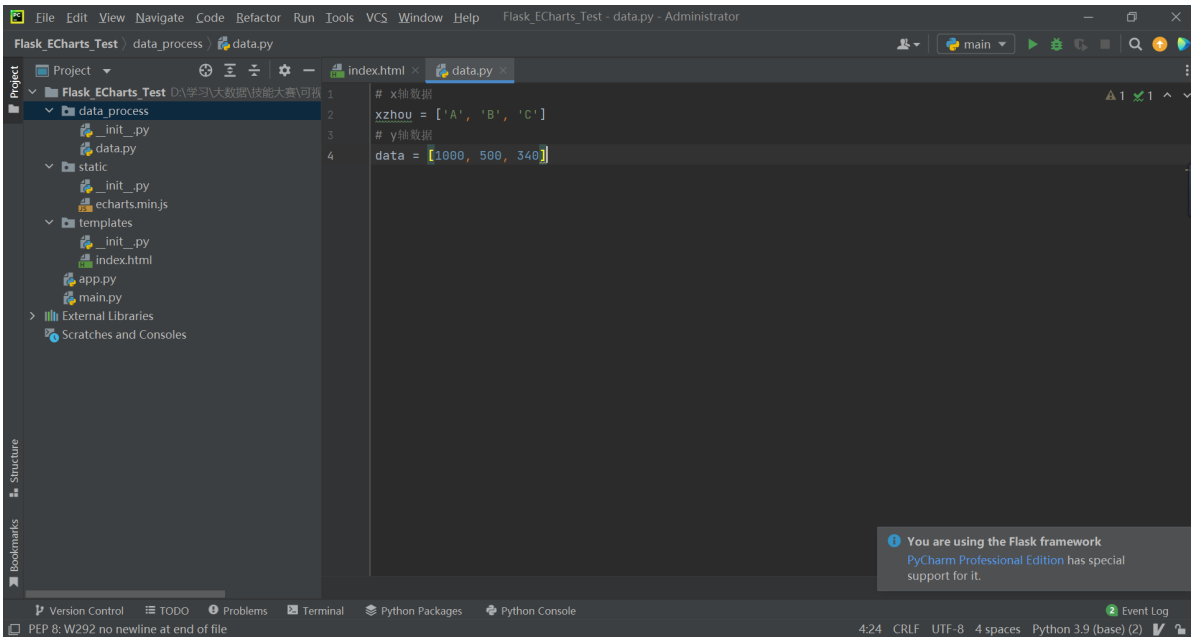
将可视化所需js脚本放入static目录中



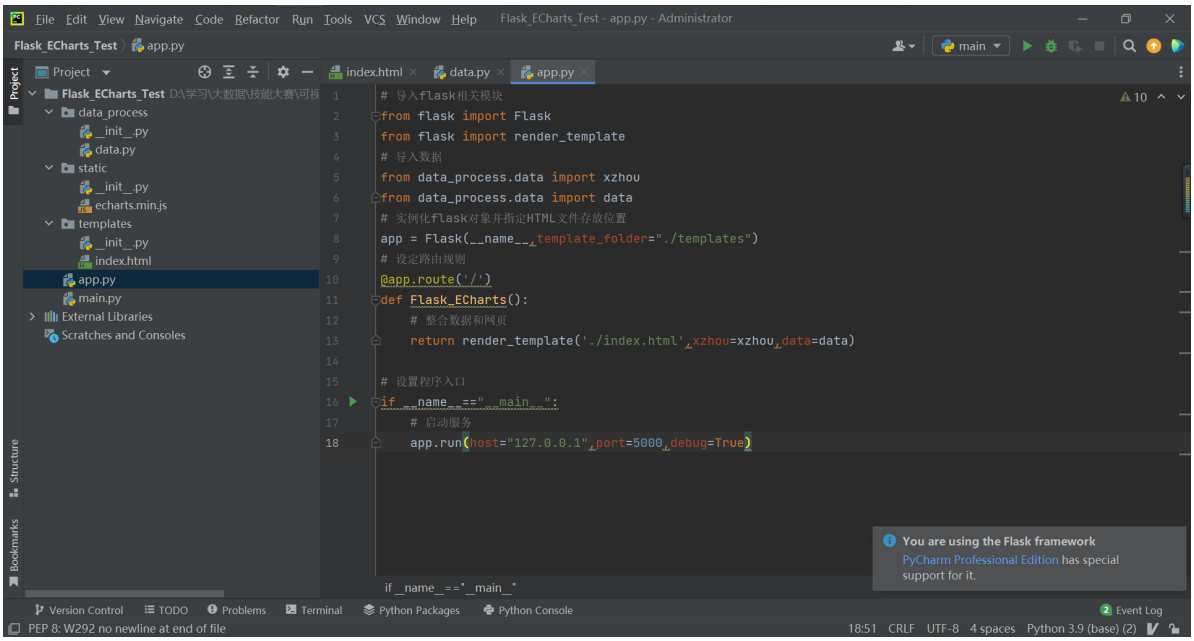
在templates下创建index.html文件



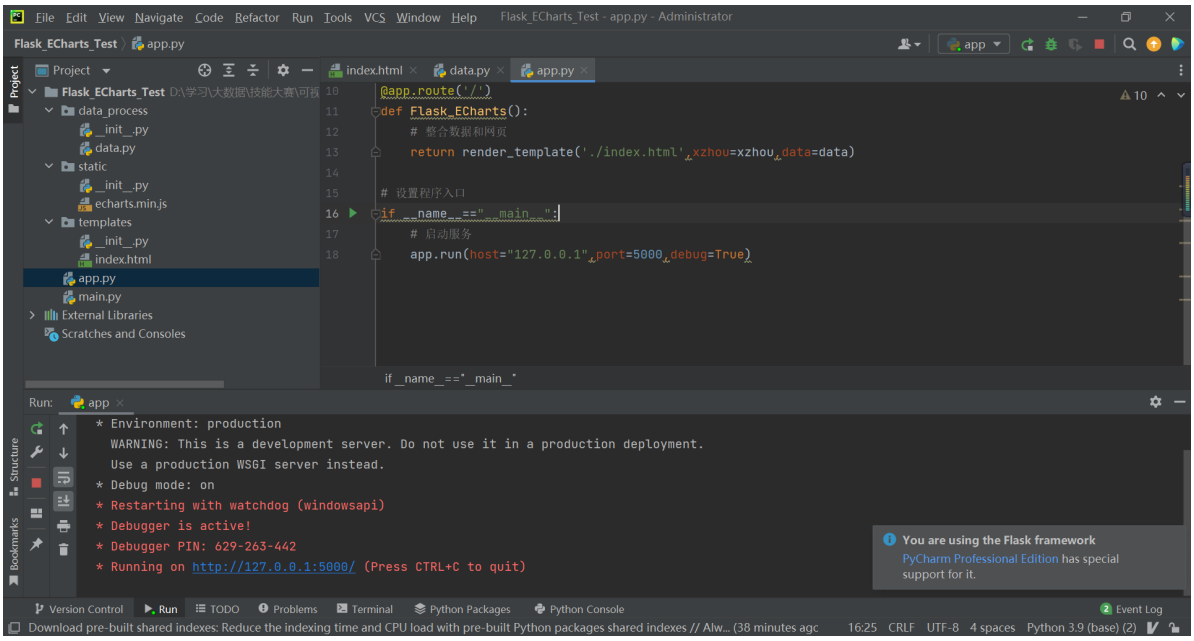
在data_process目录中编写data.py文件存放数据



在项目目录下创建app.py编写服务端代码



开启服务端



访问http://127.0.0.1:5000/