- Exercise1_Excel
  - 访问文件transactions.xlsx，获得sheet1 a1的值
  - 获取行数
  - 得到表格内的值
  - 修改表格内的值
  - 添加图表
  - 包装成函数
- Exercise2_Machine Learning
  - Libraries
  - Anaconda
    - Create Notebook
  - Kaggle
  - jupyter
    - Import CSV
    - describe()
    - values()
  - ShortCut
  - Music data
  - DecisionTree
    - import
  - Predictions
  - Calculating the Accuracy
  - model persistence
    - Load Model
  - Visualizing a Decision Tree

Exercise1_Excel

访问文件transactions.xlsx，获得sheet1 a1的值

```
import openpyxl as xl
# 把Package写作xl
wb=xl.load_workbook('transactions.xlsx')
# 加载excel文档
sheet=wb['Sheet1']
```

```
cell= sheet.cell(1,1)
# sheet1 a1 表格
print(cell.value)
# transaction_id
```

获取行数

```
import openpyxl as xl
# 把Package写作xl
wb=xl.load_workbook('transactions.xlsx')
# 加载excel文档
sheet=wb['Sheet1']
print(sheet.max_row)
# 4
```

得到表格内的值

```
import openpyxl as xl

# 把Package写作xl
wb = xl.load_workbook('transactions.xlsx')
sheet = wb['Sheet1']
# sheet1
for row in range(2, sheet.max_row + 1):
    cell = sheet.cell(row, 3)
    # (2,3)(3,3)(4,3)...(sheet.max_row,3)
     print(cell.value)
     # 得到表格内的值


# 5.95
# 6.95
# 7.95
```

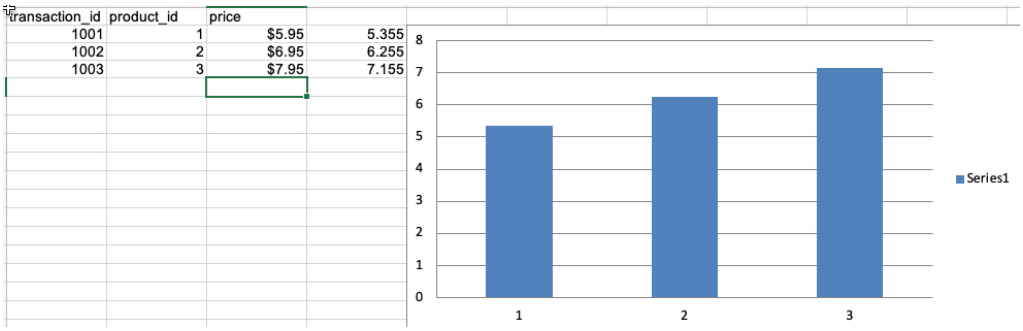
修改表格内的值

	A	B	C	D
1	transaction_id	product_id	price	
2	1001	1	$5.95	5.355
3	1002	2	$6.95	6.255
4	1003	3	$7.95	7.155

```
import openpyxl as xl

# 把Package写作xl
```

```
wb = xl.load_workbook('transactions.xlsx')
sheet = wb['Sheet1']
# sheet1
for row in range(2, sheet.max_row + 1):
    cell = sheet.cell(row, 3)
    # (2,3)(3,3)(4,3)...(sheet.max_row,3)
     corrected_price = cell.value * 0.9
     corrected_price_cell = sheet.cell(row, 4)
     # (2,4)(3,4)(4,4)...(sheet.max_row,4)
     corrected_price_cell.value = corrected_price
     # 更改表格内的值
wb.save('transactions2.xlsx')
```

添加图表




```
import openpyxl as xl
from openpyxl.chart import BarChart, Reference

# 从 openpyxl.chart导入了2个classes: BarChart,Reference
wb = xl.load_workbook('transactions.xlsx')
sheet = wb['Sheet1']
# sheet1
for row in range(2, sheet.max_row + 1):
    cell = sheet.cell(row, 3)
     # (2,3)(3,3)(4,3)...(sheet.max_row,3)
     corrected_price = cell.value * 0.9
     corrected_price_cell = sheet.cell(row, 4)
     # (2,4)(3,4)(4,4)...(sheet.max_row,4)
     corrected_price_cell.value = corrected_price
     # 更改表格内的值
values = Reference(sheet,
                   min_row=2,  # 最小行
                   max_row=sheet.max_row,
                   min_col=4,  # 最小列
                   max_col=4)
```

```
# Sheet1 (2,4)(3,4)(4,4)...(sheet.max_row,4)
chart = BarChart()
chart.add_data(values)
# 图表中添加数据
sheet.add_chart(chart, 'E2')
# 将图表添加至E2 capital E
wb.save('transactions2.xlsx')
```

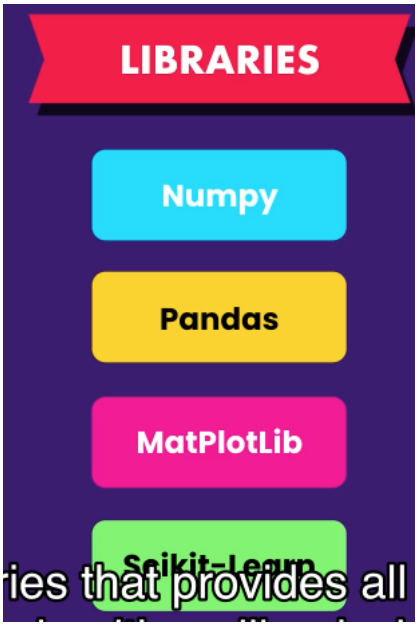
包装成函数

```
import openpyxl as xl
from openpyxl.chart import BarChart, Reference
# 从 openpyxl.chart导入了2个classes：BarChart,Reference


def process_workbook(filename):
    wb = xl.load_workbook(filename)
    sheet = wb['Sheet1']
    # sheet1
     for row in range(2, sheet.max_row + 1):
         cell = sheet.cell(row, 3)
         # (2,3)(3,3)(4,3)...(sheet.max_row,3)
         corrected_price = cell.value * 0.9
         corrected_price_cell = sheet.cell(row, 4)
         # (2,4)(3,4)(4,4)...(sheet.max_row,4)
         corrected_price_cell.value = corrected_price
         # 更改表格内的值
     values = Reference(sheet,
                        min_row=2,  # 最小行
                        max_row=sheet.max_row,
                        min_col=4,  # 最小列
                        max_col=4)
     # Sheet1 (2,4)(3,4)(4,4)...(sheet.max_row,4)
     chart = BarChart()
     chart.add_data(values)
     # 图表中添加数据
     sheet.add_chart(chart, 'E2')
     # 将图表添加至E2 capital E
     wb.save(filename)
     # overwrite
```

Exercise2_Machine Learning

4:13:00

1. Import the Data
2. Clean the Data
3. Split the Data into Training/Test Sets
4. Create a Model
5. Train the Model
6. Make Predictions
7. Evaluate and Improve



Libraries

- Numpy----Provides a multidimensional array
- Pandas----a data analysis library that provides a concept called data frame

Data frame is a two dimensional data structure similar to an excel spreadsheet电子表格 , so we have row and column

- MatPlotLib----a two dimensional plotting绘制 library for creating graphs on plots图
- Scikit-learn----provides all these common algorithms算法 like decision trees neural networks神经网络 so on

Anaconda

https://zh.wikipedia.org/wiki/Anaconda_(Python%E5%8F%91%E8%A1%8C%E7%89%88)

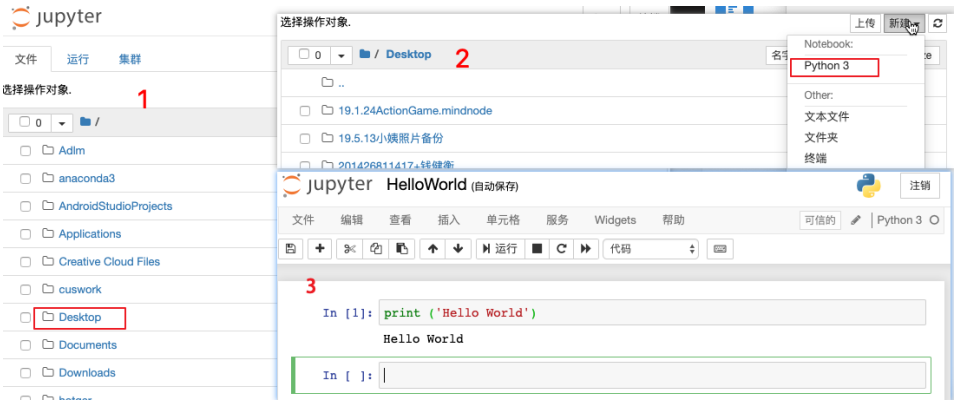
Anaconda是一个免费开源[5]的Python和R语言的发行版本，用于计算科学（数据科学、机器学习、大数据处理和预测分析），Anaconda致力于简化包管理和部署。Anaconda的包使用软件包管理系统Conda[6]进行管理。超过1200万人使用Anaconda发行版本，并且Anaconda拥有超过1400个适用于Windows、Linux和MacOS的数据科学软件包[7]。

```
$ jupyter notebook
```

启动notebook 服务

Create Notebook

简称	Jupyter
成立时间	2015
类型	nonprofit organization
目标	To support interactive data science and scientific computing across all programming languages.[1]
服务地区	Worldwide
官方语言	English
网站	jupyter.org



Kaggle

Kaggle是一个数据建模和数据分析竞赛平台。企业和研究者可在其上发布数据，统计学者和数据挖掘专家可在其上进行竞赛以产生最好的模型。这一众包模式依赖于这一事实，即有众多策略可以用于解决几乎所有预

测建模的问题，而研究者不可能在一开始就了解什么方法对于特定问题是最为有效的。Kaggle的目标则是试图通过众包的形式来解决这一难题，进而使数据科学成为一场运动。2017年3月8日谷歌官方博客宣布收购Kaggle[1]。
https://www.kaggle.com/

jupyter

Import CSV

```
import pandas as pd
df=pd.read_csv('vgsales.csv')
df
df.shape # 统计表格行数列数
 # (16598, 11)代表16598行 11列
```

```
In [2]: import pandas as pd
        df=pd.read_csv('vgsales.csv')
        df
```

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_S
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	
5	6	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	
6	7	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.38	9.23	6.50	
7	8	Wii Play	Wii	2006.0	Misc	Nintendo	14.03	9.20	2.93	
8	9	New Super Mario	Wii	2009.0	Platform	Nintendo	14.59	7.06	4.70	

describe()

```
df.describe()
#获取一些表格的基本统计信息
```

values()

```
df.values

array([[1, 'Wii Sports', 'Wii', ..., 3.77, 8.46, 82.74],
       [2, 'Super Mario Bros.', 'NES', ..., 6.81, 0.77, 40.24],
       [3, 'Mario Kart Wii', 'Wii', ..., 3.79, 3.31, 35.82],
       ...,
       [16598, 'SCORE International Baja 1000: The Official Game', 'PS2',
        ..., 0.0, 0.0, 0.01],
       [16599, 'Know How 2', 'DS', ..., 0.0, 0.0, 0.01],
       [16600, 'Spirits & Spells', 'GBA', ..., 0.0, 0.0, 0.01]],
      dtype=object)
#获取表格的基本信息
```

ShortCut

快捷键提示——按H呼出

Tab——用于呼出方法菜单

shift+Tab——用于呼出方法解释菜单

Music data

```
import pandas as pd
music_data =pd.read_csv('music.csv')
music_data
```

In [7]:
```
import pandas as pd
music_data =pd.read_csv('music.csv')
music_data
```

Out[7]:

	age	gender	genre
0	20	1	HipHop
1	23	1	HipHop
2	25	1	HipHop
3	26	1	Jazz
4	29	1	Jazz
5	30	1	Jazz
6	31	1	Classical
7	33	1	Classical
8	37	1	Classical
9	20	0	Dance
10	21	0	Dance
11	25	0	Dance
12	26	0	Acoustic
13	27	0	Acoustic
14	30	0	Acoustic
15	31	0	Classical
16	34	0	Classical
17	35	0	Classical

```
drop(columns='genre')
# 删除了'genre'列
y=music_data['genre']
# 只获取'genre'列
y

```

```
Out[11]: 0        HipHop
         1        HipHop
         2        HipHop
         3          Jazz
         4          Jazz
         5          Jazz
         6     Classical
         7     Classical
         8     Classical
         9         Dance
         10        Dance
         11        Dance
         12     Acoustic
         13     Acoustic
         14     Acoustic
         15    Classical
         16    Classical
         17    Classical
         Name: genre, dtype: object
```

DecisionTree

import

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

Predictions

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
music_data =pd.read_csv('music.csv')
X= music_data.drop(columns='genre')
Y=music_data['genre']
# 只获取'genre'列
model= DecisionTreeClassifier()
model.fit(X,Y)
predictions= model.predict([[21,1],[22,0]])
 # 预测21岁男性的喜好，22岁女性的喜好
 predictions
 # 预测21岁男性喜欢HipHop，22岁女性喜欢Dance
```

```
Out[13]: array(['HipHop', 'Dance'], dtype=object)
```

Calculating the Accuracy

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
# 导入测试类
from sklearn.metrics import accuracy_score
# 导入准确率测量类
music_data =pd.read_csv('music.csv')
X= music_data.drop(columns='genre')# 抛弃'genre'列
Y=music_data['genre']
 # 只获取'genre'列
 X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)
 # 20%的数据用来测试
```

```
model= DecisionTreeClassifier()
model.fit(X_train,Y_train)
predictions= model.predict(X_test)
# 预测X_test,Y_test
score=accuracy_score(Y_test,predictions)
# 答案与预测数据比对，得出准确率
score
```

Ctrl+Enter 快速运行

```
In [51]: X
```

Out[51]:

	age	gender
0	20	1
1	23	1
2	25	1
3	26	1
4	29	1
5	30	1
6	31	1
7	33	1
8	37	1
9	20	0
10	21	0
11	25	0
12	26	0
13	27	0
14	30	0
15	31	0
16	34	0
17	35	0

```
In [52]: Y
```

```
Out[52]: 0         HipHop
         1         HipHop
         2         HipHop
         3           Jazz
         4           Jazz
         5           Jazz
         6      Classical
         7      Classical
         8      Classical
         9          Dance
         10         Dance
         11         Dance
         12      Acoustic
         13      Acoustic
         14      Acoustic
         15     Classical
         16     Classical
         17     Classical
         Name: genre, dtype: object
```

model persistence

模型持久性/保存模型

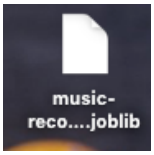
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.externals import joblib
# 导入joblib类
from sklearn.metrics import accuracy_score
# 导入准确率测量类
music_data =pd.read_csv('music.csv')
X= music_data.drop(columns='genre')# 抛弃'genre'列
Y=music_data['genre']
 # 只获取'genre'列

 joblib.dump(model,'music-recommender.joblib')
```

模型文件保存成功

```
joblib.dump(model,'music-recommender.joblib')
```

```
Out[384]: ['music-recommender.joblib']
```

桌面



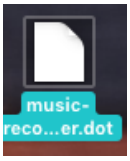
Load Model

```
model=joblib.load('music-recommender.joblib')
```

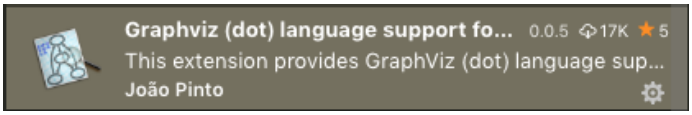
```
# 加载之前保存的model文件
predictions= model.predict([[21,1]])
predictions
# 进行预测

# array(['HipHop'], dtype=object)
```

Visualizing a Decision Tree

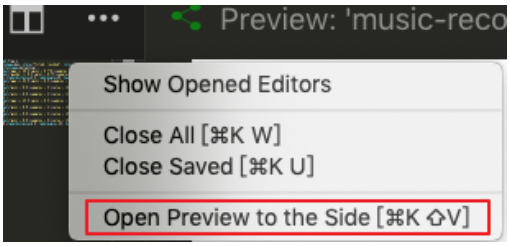
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
# 导入tree类

music_data =pd.read_csv('music.csv')
X= music_data.drop(columns='genre')# 抛弃'genre'列
Y=music_data['genre']
# 只获取'genre'列
 model= DecisionTreeClassifier()
 model.fit(X,Y)

 tree.export_graphviz(model,out_file='music-recommender.dot',
                      feature_names=['age','gender'],
                      class_names=sorted(Y.unique()),
                      label='all',
                      rounded=True,
                      filled=True)
# 生成.dot机器学习流程图
```

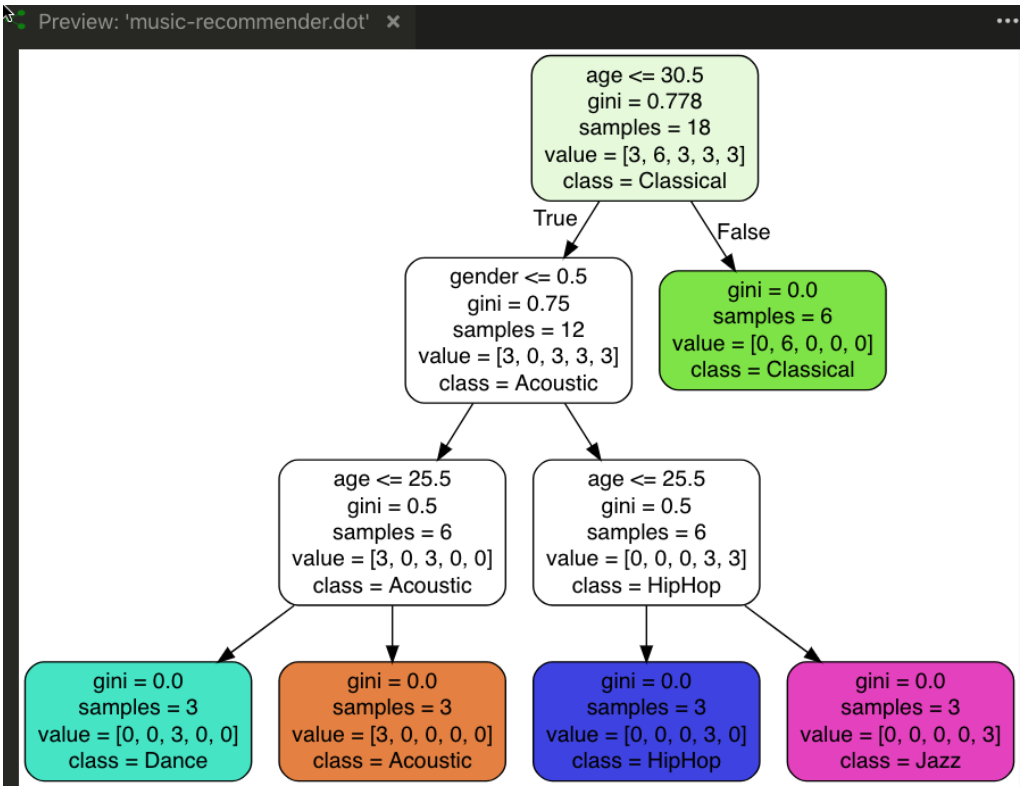


Visual Studio Code



安装此扩展插件






```
tree.export_graphviz(model,out_file='music-recommender.dot',
                     feature_names=['age','gender'],
#                       so we can see the rules in our node
                     class_names=sorted(Y.unique()),
#                       displaying the class for each node
                     label='all',
#                       every node label we can read
                     rounded=True,
#                       rounded corners
                      filled=True
                       # each box/node is filled with a color
                      )
```

Exercise3_Web Site

Django

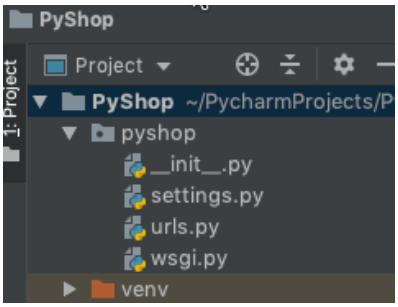
framework——a library of reusable modules

Install Django

```
$pip install django==2.1
```

Create Project

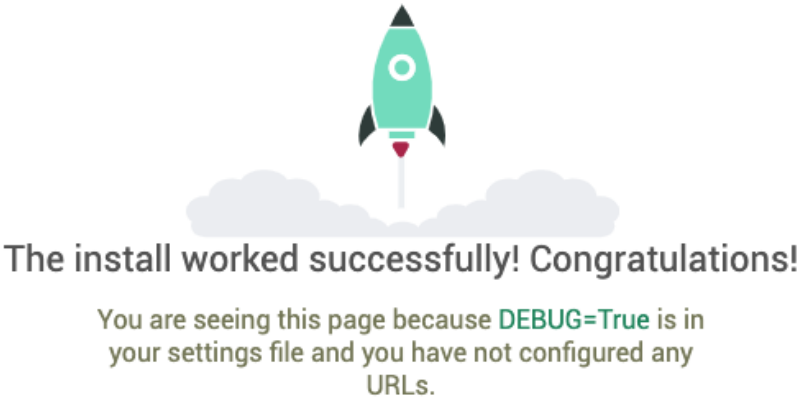
```
$ django-admin startproject pyshop .
# '.' current folder
```



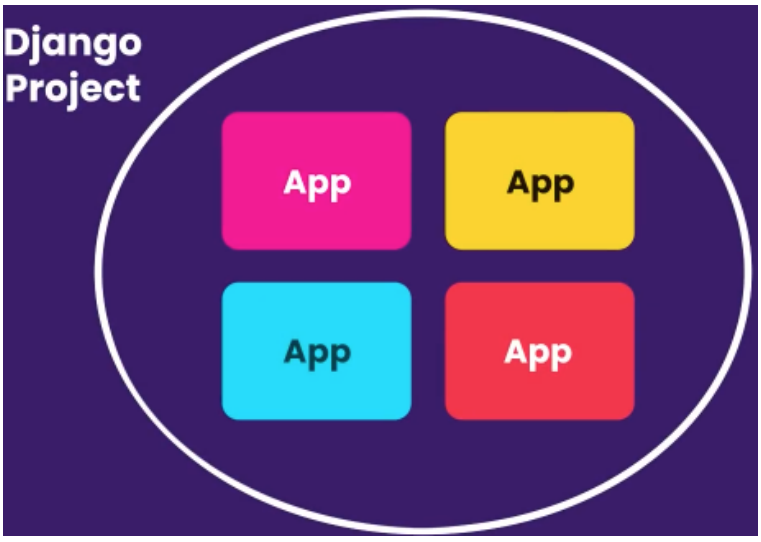
Starting development server

```
$ python manage.py runserver
```

http://127.0.0.1:8000/



django运行成功



View Function

views.py

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

```python
# /products ->index
# Uniform Resource Locator (Address)
def index(request):
    return HttpResponse('Hello world')
```

Map Urls

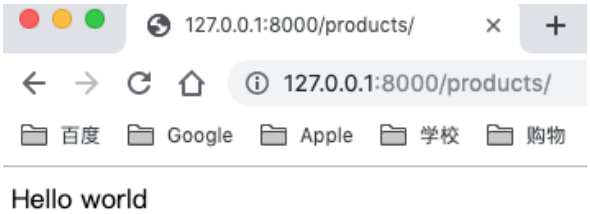
products/urls.py

```python
from django.urls import path
from . import views

# '.' means current folder
# /products
# /products/1/detail
# /products/new
urlpatterns = [
    path('', views.index)
     #     map urls to this views function
]
```

pyshop/urls.py

```python
from django.contrib import admin
from django.urls import path, include

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

匹配网址成功



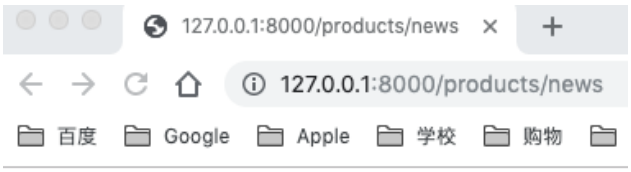
为http://127.0.0.1:8000/products/news添加匹配

views.py

```python
def news(request):
    return HttpResponse('New Products')
```

products/urls.py

```python
path('news', views.news)
```



New Products

Model

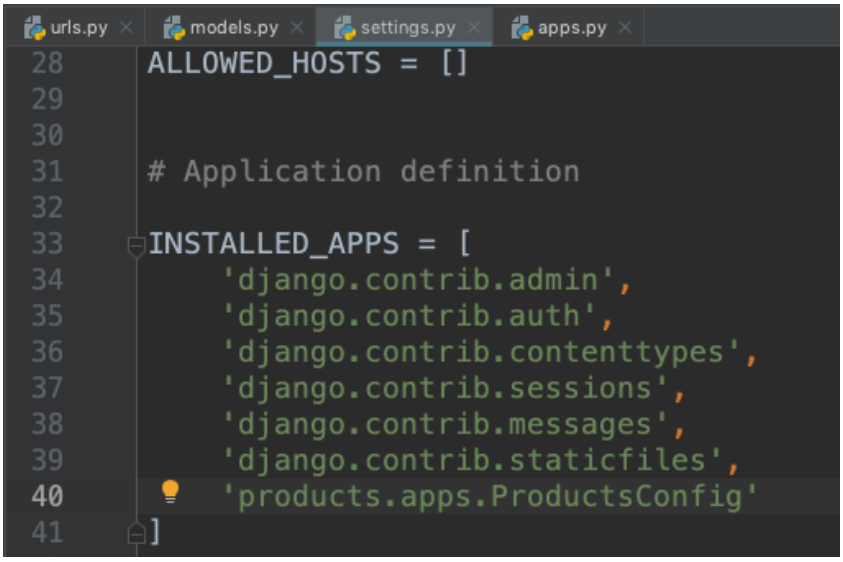
Database——DB Browser for SQLite

Migration

```
$ python3 manage.py makemigrations
```

```
(venv) (base) MorrisdeMacBook-Pro:PyShop morris$ python3 manage.py makemigrations
No changes detected
```

添加这句话



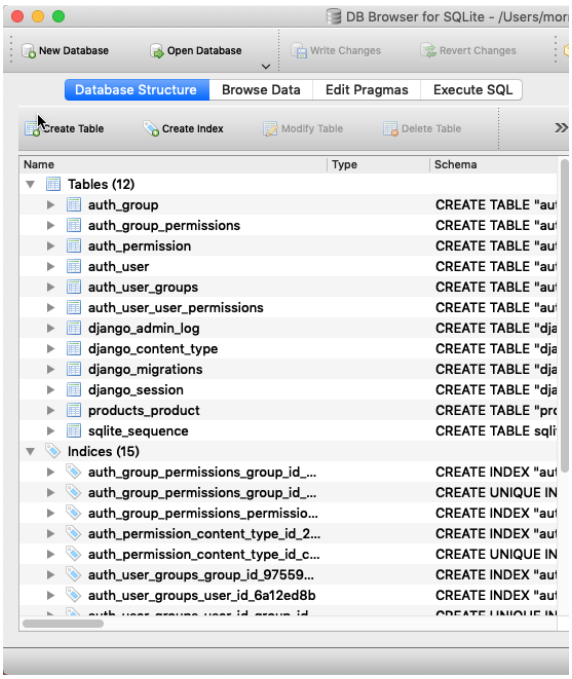

```
(venv) (base) MorrisdeMacBook-Pro:PyShop morris$ python3 manage.py makemigrations
Migrations for 'products':
  products/migrations/0001_initial.py
    - Create model Product
```

Database

DB Browser for SQLite

Generate Product Table

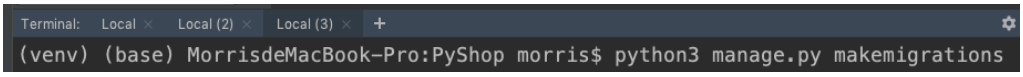
```
$ python3 manage.py migrate
```



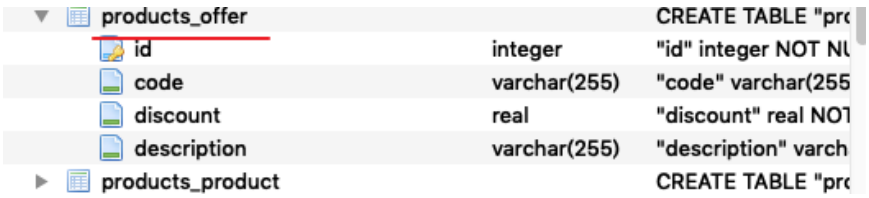
添加新类，数据库更新

models.py

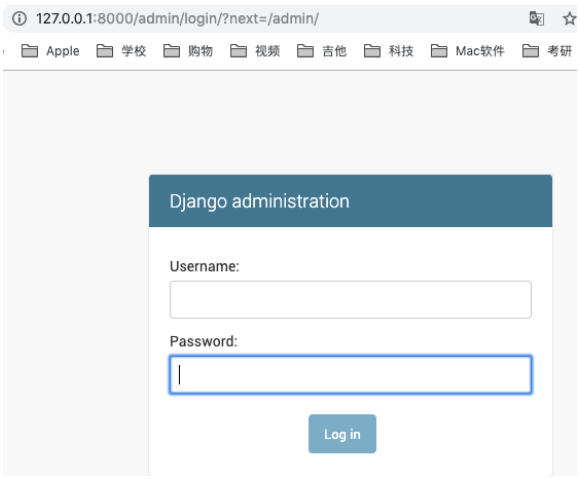
```
class Offer(models.Model):
    code = models.CharField(max_length=255)
    description = models.CharField(max_length=255)
    discount = models.FloatField()
```




```
python3 manage.py makemigrations
python3 manage.py migrate

```

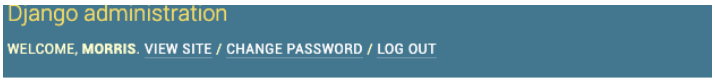


Create SuperAdmin

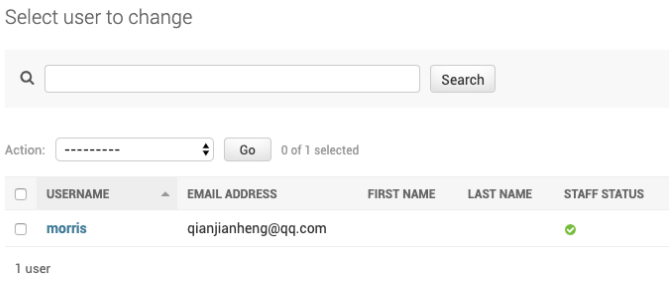



```
python3 manage.py createsuperuser
```

使用超级管理员登陆



用户管理



Create Products

admin.py

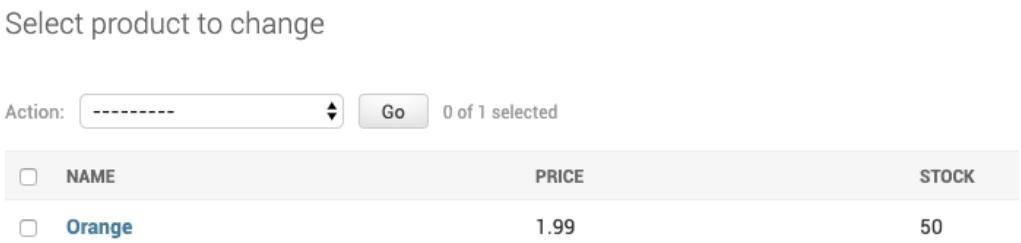
```
from django.contrib import admin
from .models import Product

admin.site.register(Product)
```

Customizing the Admin

admin.py

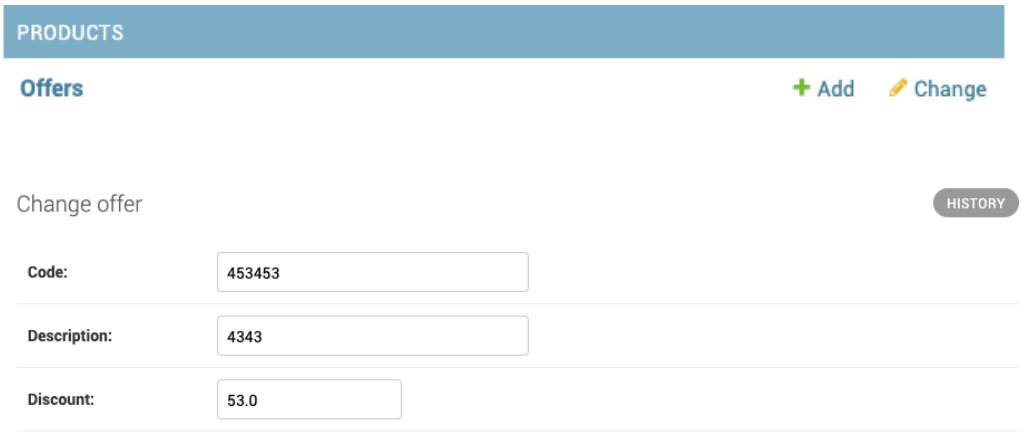
```
class ProductAdmin(admin.ModelAdmin):
```

```
    list_display = ('name', 'price', 'stock')

    admin.site.register(Product, ProductAdmin)
```



Add Offers Table

```
from .models import Offer

class OfferAdmin(admin.ModelAdmin):
    list_display = ('code', 'discount')
admin.site.register(Offer, OfferAdmin)
```



Let Products Show on the Website

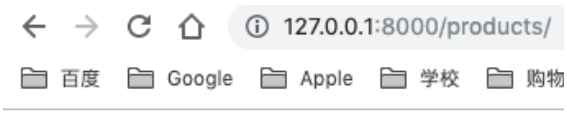
views.py

```
from .models import Product

# /products ->index
# Uniform Resource Locator (Address)
def index(request):
    return HttpResponse('Hello world')


def news(request):
     products=Product.objects.all()
     return HttpResponse('New Products')
```

Create directory->Create HTML File


```
def index(request):
    products = Product.objects.all()
    return render(request, 'index.html')
```

http://127.0.0.1:8000/products/



Products

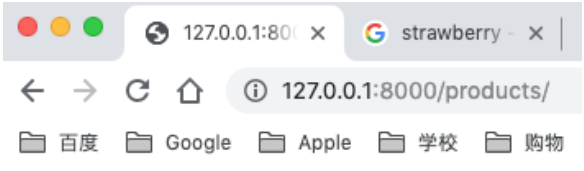
- Item 1
- Item 2
- Item 3

HTML file

```
<h1>Products</h1>
<ul>
    {% for product in products %}
<!--    使得可以执行Python语言-->
        <li>{{product.name}}</li>
    {% endfor %}
</ul>
```

views.py

```
def index(request):
    # return HttpResponse('Hello world')
    products = Product.objects.all()
    return render(request, 'index.html',
                  {'products': products})
```



Products

- Orange
- Strawberry

index.html

```
<h1>Products</h1>
<ul>
    {% for product in products %}
<!--    使得可以执行Python语言-->
        <li>{{product.name}}({{product.price}})</li>
<!--    显示名称，（价格）-->
    {% endfor %}
</ul>
```

Products

- Orange(1.99)
- Strawberry(2.99)

Adding Bootstrap

引导程序

Bootstrap 是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目。

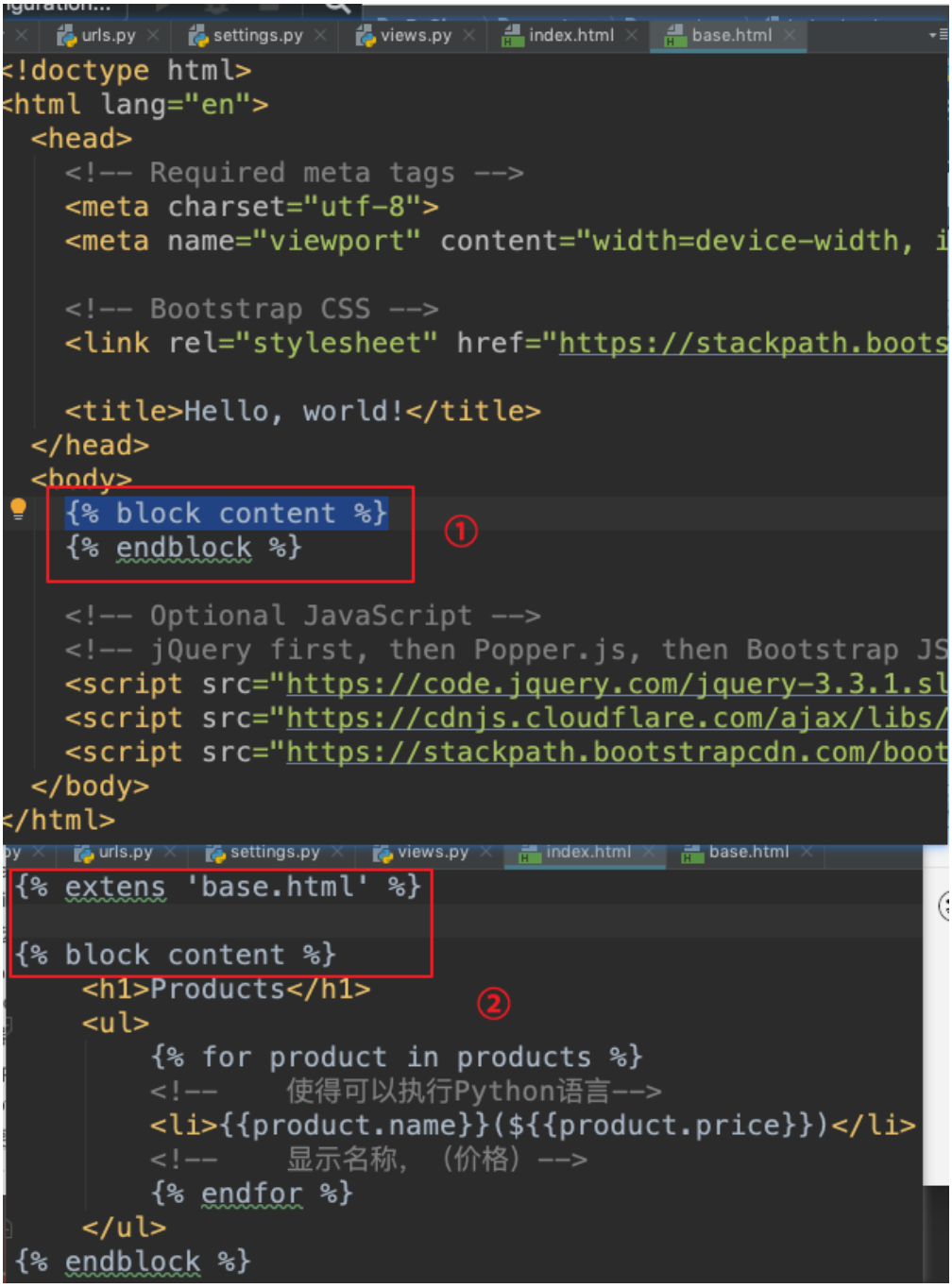
https://getbootstrap.com/docs/4.3/getting-started/introduction/

base.html

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-f

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1

     <title>Hello, world!</title>
  </head>
  <body>
    {% block content %}
    {% endblock %}

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha3
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/poppe
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.m
  </body>
</html>
```

index.html

```
1  {% extens 'base.html' %}
```



样式变好看了

Products

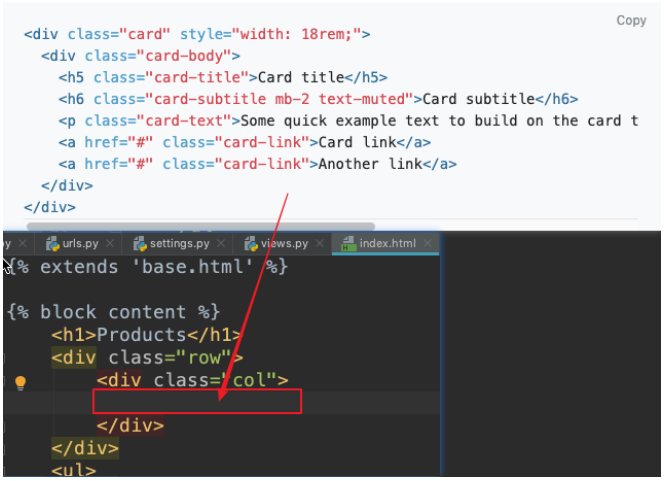
- Orange($1.99)
- Strawberry($2.99)

Rendering Cards

https://getbootstrap.com/docs/4.3/components/card/

Cards

Bootstrap's cards provide a flexible and extensible content container with multiple variants and options.

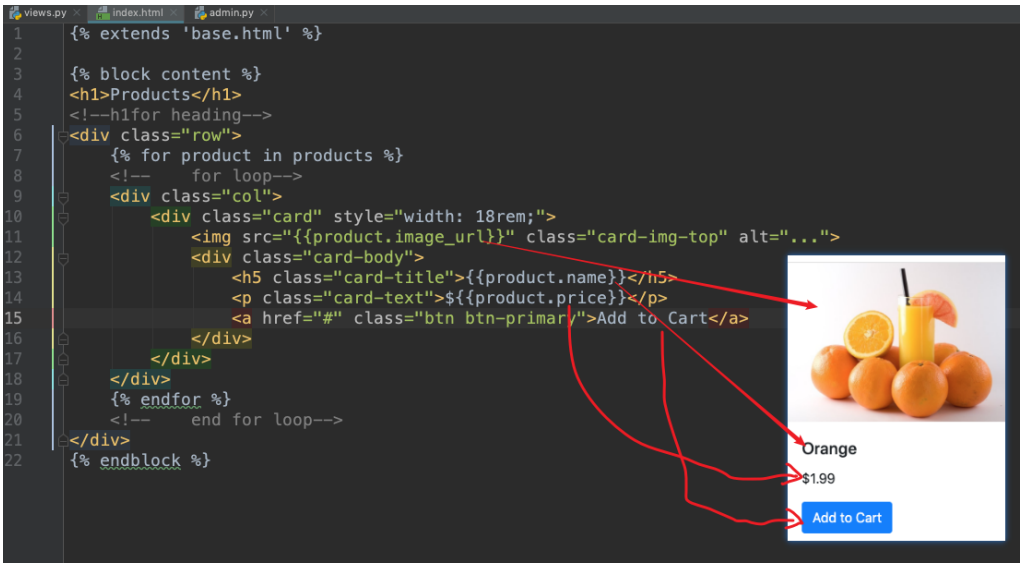


row-col-card

```
{% extends 'base.html' %}

{% block content %}
    <h1>Products</h1>
    <!--h1for heading-->
    <div class="row">
        {% for product in products %}
        <!--    for loop-->
        <div class="col">
            <div class="card" style="width: 18rem;">
                <div class="card-body">
                    <h5 class="card-title">Card title</h5>
                    <h6 class="card-subtitle mb-2 text-muted">Card subtitle</h6>
                    <p class="card-text">Some quick example text to build on the c
                        card's content.</p>
                    <a href="#" class="card-link">Card link</a>
                    <a href="#" class="card-link">Another link</a>
                </div>
            </div>
        </div>
        {% endfor %}
```

```
22          <!--    end for loop-->
23      </div>
24  {% endblock %}
```

Add Image



Add Navigation Bar

Brand Navbar

The `.navbar-brand` can be applied to most elements, but an anchor works best as some elements might require utility classes or custom styles.

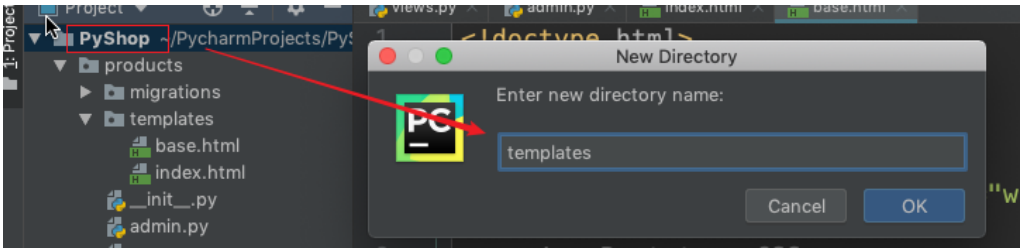
Navbar

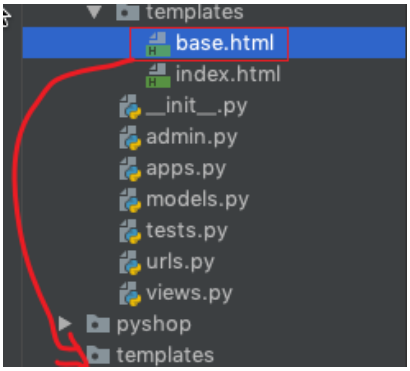
Navbar

Copy

```
<!-- As a link -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
</nav>
```

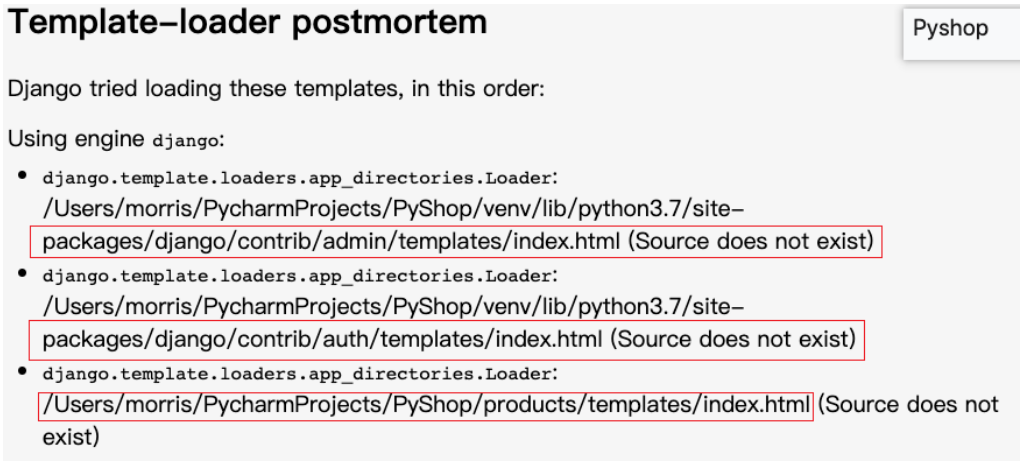
Reuse it on multiple apps



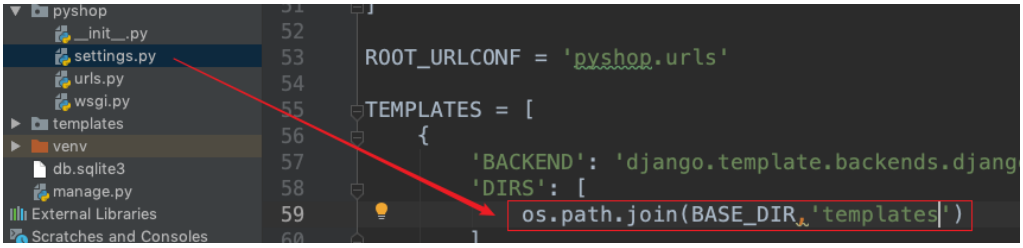


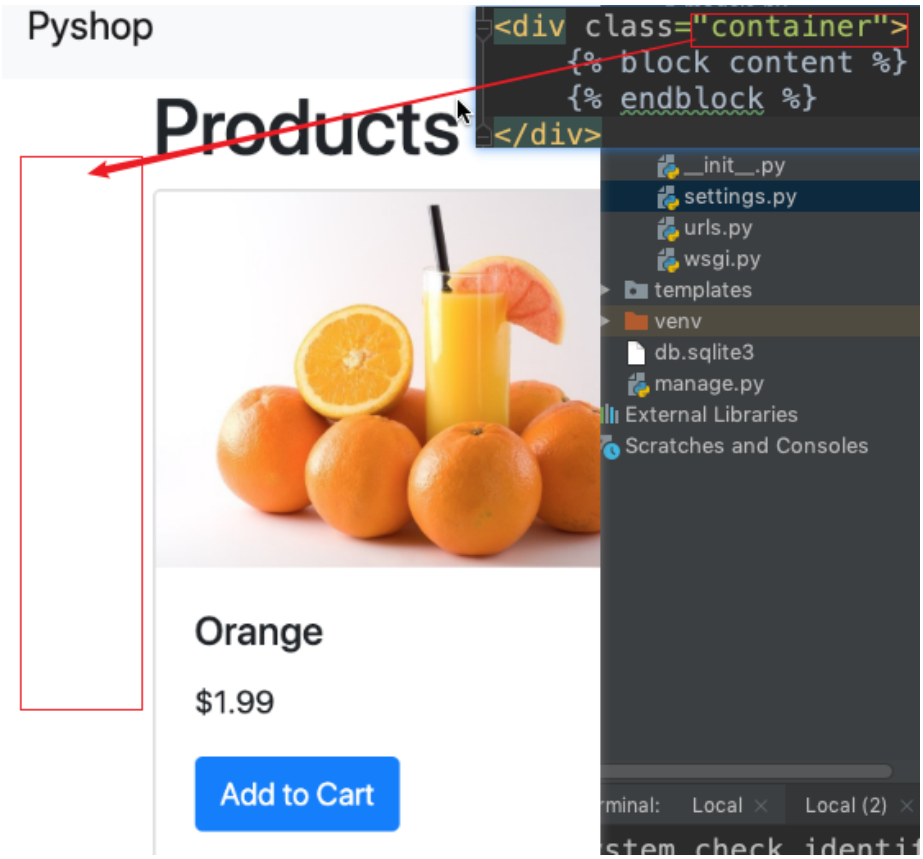
移动后访问http://127.0.0.1:8000/products/

出现错误



添加这一句




Pyshop
Products

{% block content %}
{% endblock %}

__init__.py
settings.py
urls.py
wsgi.py
templates
venv
db.sqlite3
manage.py
External Libraries
Scratches and Consoles
Orange
$1.99
Add to Cart
Local
Local (2)