

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

Exercise3_Web Site

Django

Install Django

Create Project

View Function

Map Urls

Model

Migration

Database

Generate Product Table

Create SuperAdmin

Create Products

Customizing the Admin

Add Offers Table

Let Products Show on the Website

HTML file

Adding Bootstrap

Rendering Cards

Add Image

Add Navigation Bar

Reuse it on multiple apps

Exercise1_Excel

访问文件transactions.xlsx，获得sheet1 a1的值

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

```

6 cell= sheet.cell(1,1)
7 # sheet1 a1 表格
8 print(cell.value)
9 # transaction_id

```

获取行数

```

1 import openpyxl as xl
2 # 把Package写作xl
3 wb=xl.load_workbook('transactions.xlsx')
4 # 加载excel文档
5 sheet=wb['Sheet1']
6 print(sheet.max_row)
7 # 4

```

得到表格内的值

```

1 import openpyxl as xl
2
3 # 把Package写作xl
4 wb = xl.load_workbook('transactions.xlsx')
5 sheet = wb['Sheet1']
6 # sheet1
7 for row in range(2, sheet.max_row + 1):
8     cell = sheet.cell(row, 3)
9     # (2,3)(3,3)(4,3)...(sheet.max_row,3)
10    print(cell.value)
11    # 得到表格内的值
12
13
14 # 5.95
15 # 6.95
16 # 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

```

1 import openpyxl as xl
2
3 # 把Package写作xl

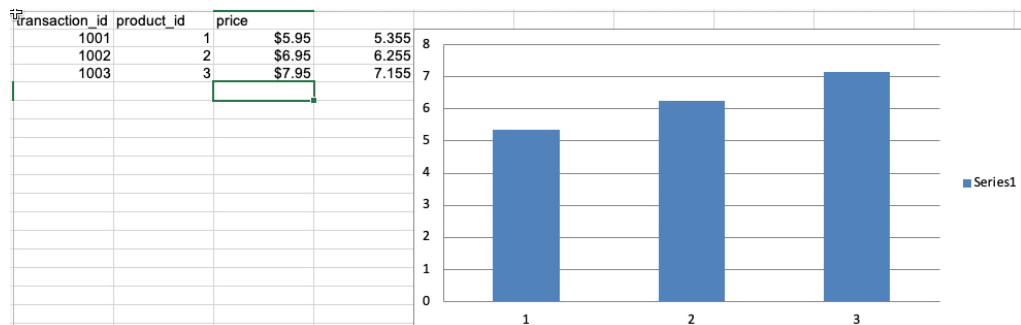
```

```

4 wb = xl.load_workbook('transactions.xlsx')
5 sheet = wb['Sheet1']
6 # sheet1
7 for row in range(2, sheet.max_row + 1):
8     cell = sheet.cell(row, 3)
9     # (2,3)(3,3)(4,3)...(sheet.max_row,3)
10    corrected_price = cell.value * 0.9
11    corrected_price_cell = sheet.cell(row, 4)
12    # (2,4)(3,4)(4,4)...(sheet.max_row,4)
13    corrected_price_cell.value = corrected_price
14    # 更改表格内的值
15 wb.save('transactions2.xlsx')

```

添加图表



```

1 import openpyxl as xl
2 from openpyxl.chart import BarChart, Reference
3
4 # 从 openpyxl.chart 导入了 2 个 classes: BarChart, Reference
5 wb = xl.load_workbook('transactions.xlsx')
6 sheet = wb['Sheet1']
7 # sheet1
8 for row in range(2, sheet.max_row + 1):
9     cell = sheet.cell(row, 3)
10    # (2,3)(3,3)(4,3)...(sheet.max_row,3)
11    corrected_price = cell.value * 0.9
12    corrected_price_cell = sheet.cell(row, 4)
13    # (2,4)(3,4)(4,4)...(sheet.max_row,4)
14    corrected_price_cell.value = corrected_price
15    # 更改表格内的值
16 values = Reference(sheet,
17                     min_row=2, # 最小行
18                     max_row=sheet.max_row,
19                     min_col=4, # 最小列
20                     max_col=4)

```

```

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

```

包装成函数

```

1 import openpyxl as xl
2 from openpyxl.chart import BarChart, Reference
3 # 从 openpyxl.chart导入了2个classes: BarChart,Reference
4
5
6 def process_workbook(filename):
7     wb = xl.load_workbook(filename)
8     sheet = wb['Sheet1']
9     # sheet1
10    for row in range(2, sheet.max_row + 1):
11        cell = sheet.cell(row, 3)
12        # (2,3)(3,3)(4,3)...(sheet.max_row,3)
13        corrected_price = cell.value * 0.9
14        corrected_price_cell = sheet.cell(row, 4)
15        # (2,4)(3,4)(4,4)...(sheet.max_row,4)
16        corrected_price_cell.value = corrected_price
17        # 更改表格内的值
18    values = Reference(sheet,
19                        min_row=2, # 最小行
20                        max_row=sheet.max_row,
21                        min_col=4, # 最小列
22                        max_col=4)
23    # Sheet1 (2,4)(3,4)(4,4)...(sheet.max_row,4)
24    chart = BarChart()
25    chart.add_data(values)
26    # 图表中添加数据
27    sheet.add_chart(chart, 'E2')
28    # 将图表添加至E2 capital E
29    wb.save(filename)
30    # overwrite

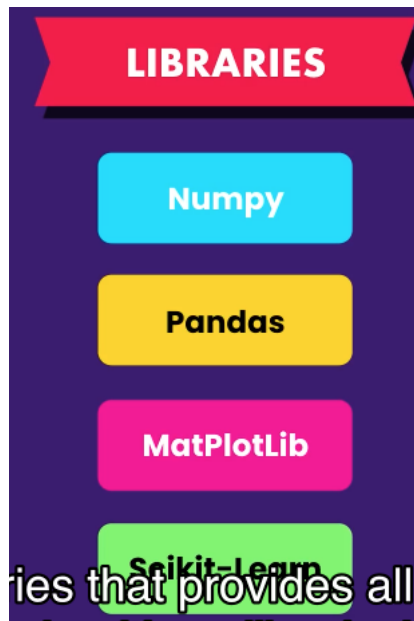
```

Exercise2_Machine Learning

4:13:00

Steps

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_\(Python%E5%8F%91%E8%A1%8C%E7%89%88\)](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]。

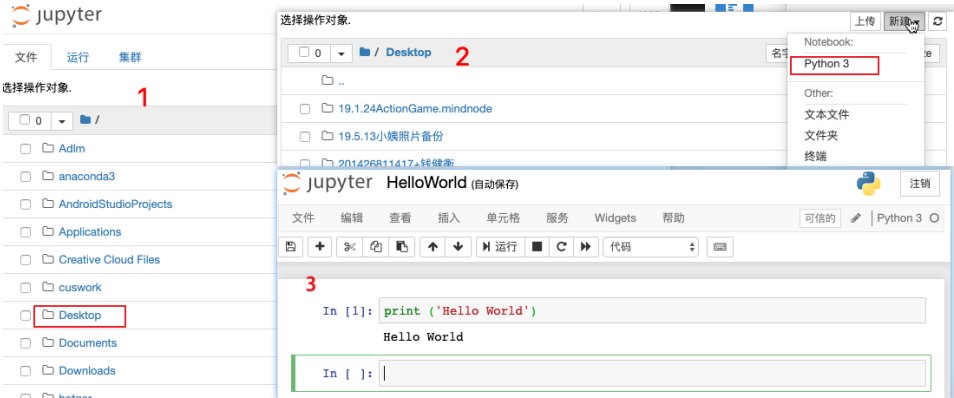
```
1 $ jupyter notebook
```

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

```
1 import pandas as pd
2 df=pd.read_csv('vgsales.csv')
3 df
4 df.shape # 统计表格行数列数
5 # (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	2006.0	Platform	Nintendo	14.50	7.08	4.70	

describe()

```
1 df.describe()
2 #获取一些表格的基本统计信息
```

values()

```
1 df.values
2
3 array([[1, 'Wii Sports', 'Wii', ..., 3.77, 8.46, 82.74],
4        [2, 'Super Mario Bros.', 'NES', ..., 6.81, 0.77, 40.24],
5        [3, 'Mario Kart Wii', 'Wii', ..., 3.79, 3.31, 35.82],
6        ...,
7        [16598, 'SCORE International Baja 1000: The Official Game', 'PS2',
8         ..., 0.0, 0.0, 0.01],
9        [16599, 'Know How 2', 'DS', ..., 0.0, 0.0, 0.01],
10       [16600, 'Spirits & Spells', 'GBA', ..., 0.0, 0.0, 0.01]],
11      dtype=object)
12 #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

```
1 import pandas as pd
2 music_data =pd.read_csv('music.csv')
3 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

```
1 drop(columns='genre')
2 # 删除了'genre'列
3 y=music_data['genre']
4 # 只获取'genre'列
5 y
6
```

```

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

```

1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier

```

Predictions

```

1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 music_data =pd.read_csv('music.csv')
4 X= music_data.drop(columns='genre')
5 Y=music_data['genre']
6 # 只获取'genre'列
7 model= DecisionTreeClassifier()
8 model.fit(X,Y)
9 predictions= model.predict([[21,1],[22,0]])
10 # 预测21岁男性的喜好, 22岁女性的喜好
11 predictions
12 # 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

```

1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.model_selection import train_test_split
4 # 导入测试类
5 from sklearn.metrics import accuracy_score
6 # 导入准确率测量类
7 music_data =pd.read_csv('music.csv')
8 X= music_data.drop(columns='genre')# 抛弃'genre'列
9 Y=music_data['genre']
10 # 只获取'genre'列
11 X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)
12 # 20%的数据用来测试

```

```
13 model= DecisionTreeClassifier()
14 model.fit(X_train,Y_train)
15 predictions= model.predict(X_test)
16 # 预测X_test,Y_test
17 score=accuracy_score(Y_test,predictions)
18 # 答案与预测数据比对, 得出准确率
19 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

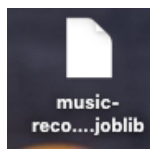
```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.externals import joblib
4 # 导入joblib类
5 from sklearn.metrics import accuracy_score
6 # 导入准确率测量类
7 music_data =pd.read_csv('music.csv')
8 X= music_data.drop(columns='genre')# 抛弃'genre'列
9 Y=music_data['genre']
10 # 只获取'genre'列
11
12 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

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

```

2 # 加载之前保存的model文件
3 predictions= model.predict([[21,1]])
4 predictions
5 # 进行预测
6
7 # array(['HipHop'], dtype=object)

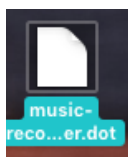
```

Visualizing a Decision Tree

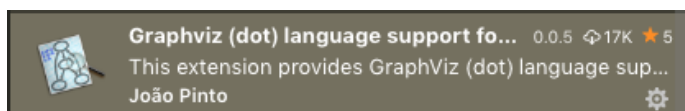
```

1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn import tree
4 # 导入tree类
5
6 music_data =pd.read_csv('music.csv')
7 X= music_data.drop(columns='genre')# 抛弃'genre'列
8 Y=music_data['genre']
9 # 只获取'genre'列
10 model= DecisionTreeClassifier()
11 model.fit(X,Y)
12
13 tree.export_graphviz(model,out_file='music-recommender.dot',
14                       feature_names=['age', 'gender'],
15                       class_names=sorted(Y.unique()),
16                       label='all',
17                       rounded=True,
18                       filled=True)
19 # 生成.dot机器学习流程图

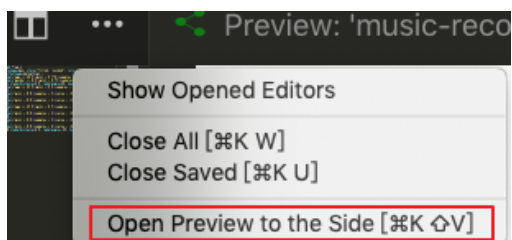
```

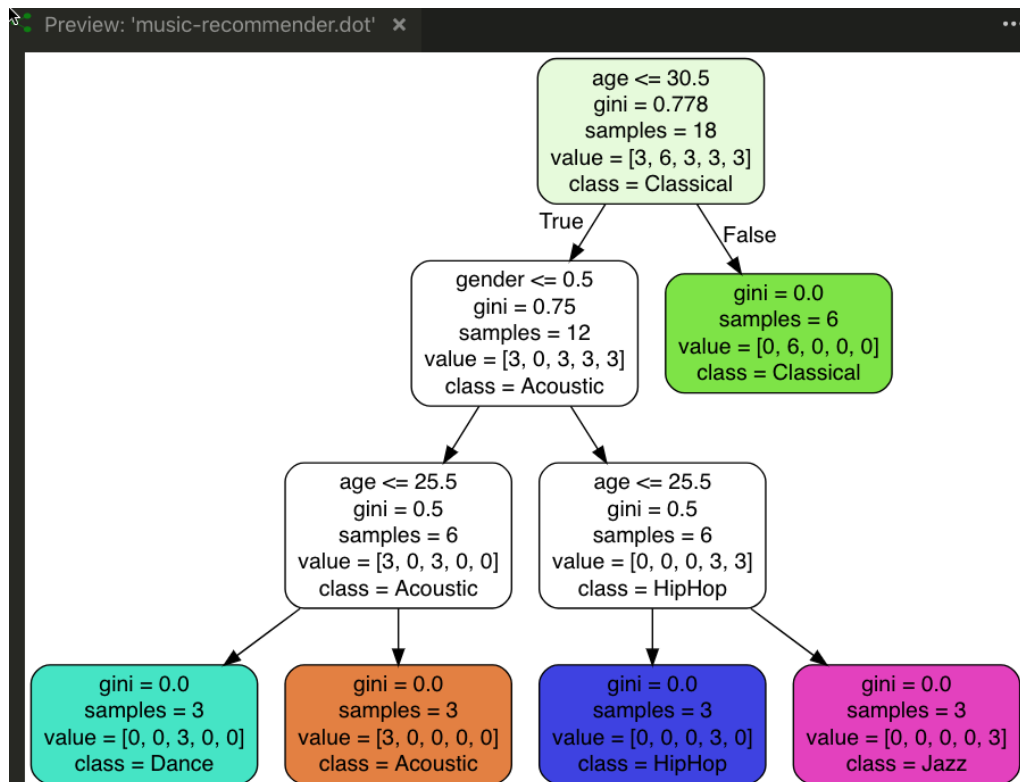


Visual Studio Code



安装此扩展插件





```

1 tree.export_graphviz(model,out_file='music-recommender.dot',
2                       feature_names=['age', 'gender'],
3 #                       so we can see the rules in our node
4                       class_names=sorted(Y.unique()),
5 #                       displaying the class for each node
6                       label='all',
7 #                       every node label we can read
8                       rounded=True,
9 #                       rounded corners
10                      filled=True
11                      # each box/node is filled with a color
12                      )

```

Exercise3_Web Site

Django

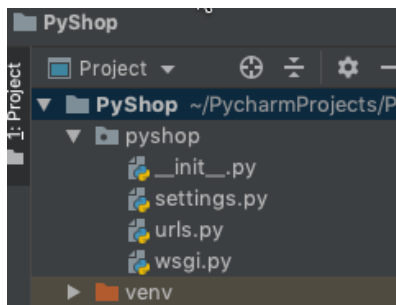
framework—a library of reusable modules

Install Django

```
1 $pip install django==2.1
```

Create Project

```
1 $ django-admin startproject pyshop .
2 # '.' current folder
```



Starting development server

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

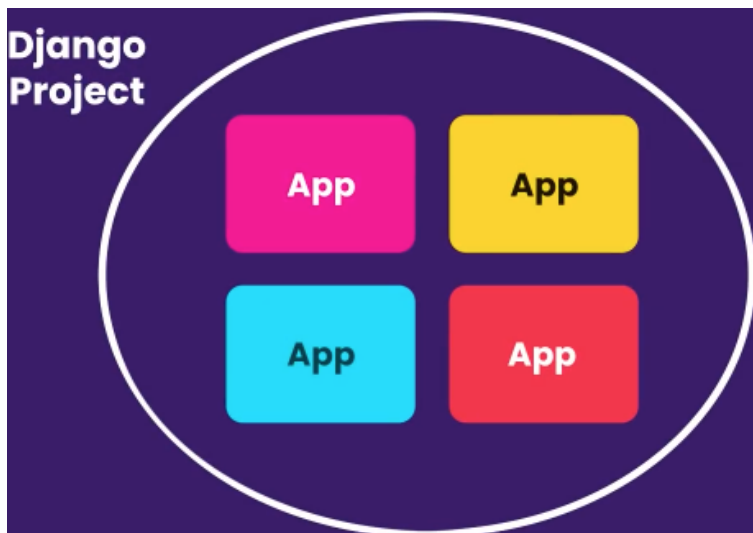
<http://127.0.0.1:8000/>



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

django运行成功



View Function

views.py

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

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

Map Urls

urls.py

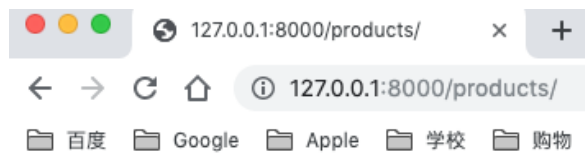
products/urls.py

```
1 from django.urls import path
2 from . import views
3
4 # '.' means current folder
5 # /products
6 # /products/1/detail
7 # /products/new
8 urlpatterns = [
9     path('', views.index)
10    #     map urls to this views function
11 ]
```

pyshop/urls.py

```
1 from django.contrib import admin
2 from django.urls import path, include
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('products/', include('products.urls'))
7 ]
```

匹配网址成功



Hello world

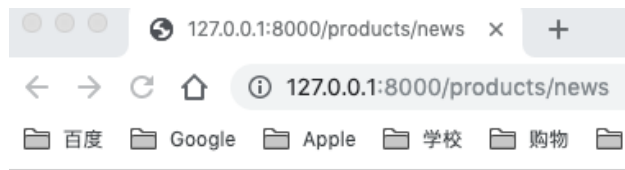
为http://127.0.0.1:8000/products/news添加匹配

views.py

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

products/urls.py

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

New Products

Model

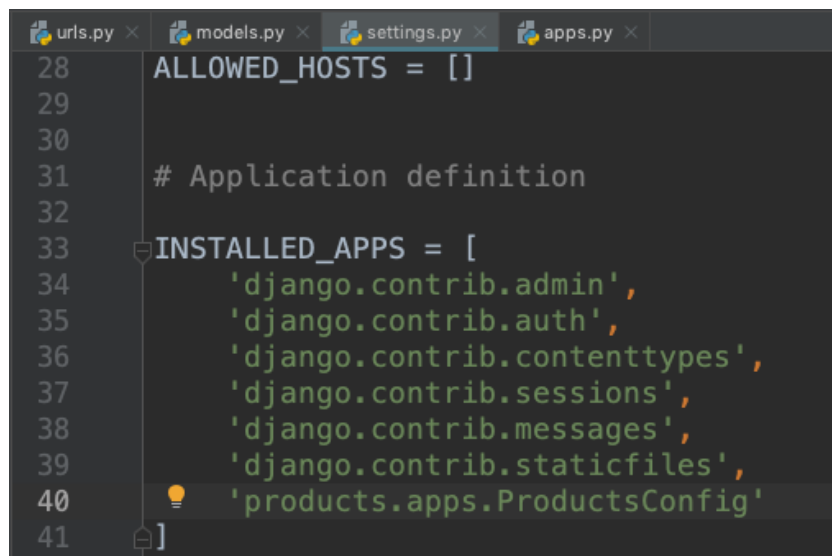
Database——DB Browser for SQLite

Migration

```
1 $ python3 manage.py makemigrations
```

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

添加这句话



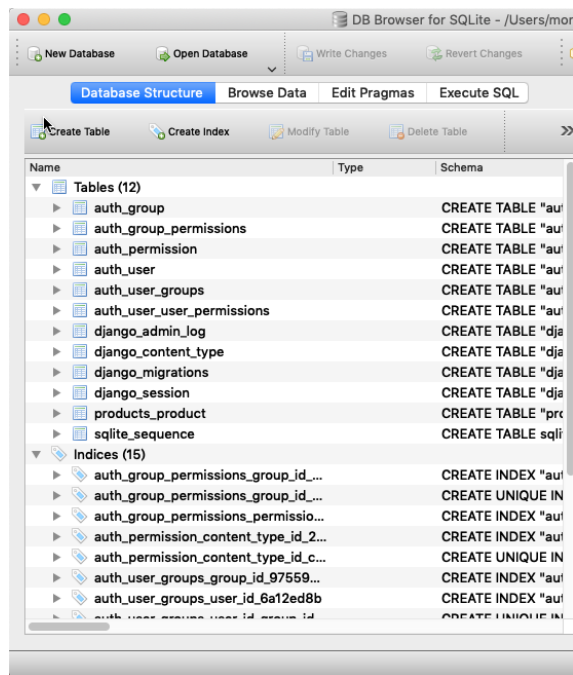
```
(venv) (base) MorrisdeMacBook-Pro:PyShop morris$ python3 manage.py makemigrations
Migrations for 'products':
  products/migrations/0001_initial.py
    - Create model Product
```

Database

DB Browser for SQLite

Generate Product Table

```
1 $ python3 manage.py migrate
```



Name	Type	Schema
products_product		CREATE TABLE "prc"
id	integer	"id" integer NOT NU
name	varchar(255)	"name" varchar(255
price	real	"price" real NOT NU
stock	integer	"stock" integer NO
image_url	varchar(2083)	"image_url" varchar

添加新类，数据库更新

models.py

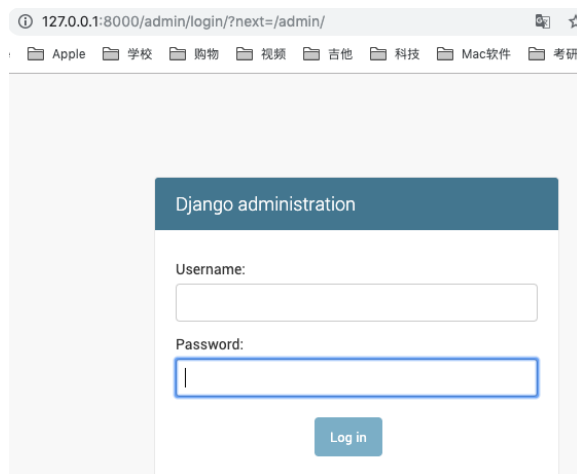
```
1 class Offer(models.Model):
2     code = models.CharField(max_length=255)
3     description = models.CharField(max_length=255)
4     discount = models.FloatField()
```

```
Terminal: Local x Local (2) x Local (3) x +
(venv) (base) MorrisdeMacBook-Pro:PyShop morris$ python3 manage.py makemigrations
```

```
1 python3 manage.py makemigrations
2 python3 manage.py migrate
3
```

Name	Type	Schema
products_offer		CREATE TABLE "prc"
id	integer	"id" integer NOT NU
code	varchar(255)	"code" varchar(255
discount	real	"discount" real NO
description	varchar(255)	"description" varch
products_product		CREATE TABLE "prc"

Create SuperAdmin



```
1 python3 manage.py createsuperuser
```

使用超级管理员登陆



Site administration

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add Change
Users	+ Add Change

用户管理

Select user to change

Action: 0 of 1 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	morris	qianjianheng@qq.com			<input checked="" type="checkbox"/>

1 user

Create Products

admin.py

```
1 from django.contrib import admin
2 from .models import Product
3
4 admin.site.register(Product)
```

Customizing the Admin

admin.py

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

```

2     list_display = ('name', 'price', 'stock')
3
4     admin.site.register(Product, ProductAdmin)

```

Select product to change

Action: 0 of 1 selected

<input type="checkbox"/>	NAME	PRICE	STOCK
<input type="checkbox"/>	Orange	1.99	50

Add Offers Table

```

1 from .models import Offer
2
3 class OfferAdmin(admin.ModelAdmin):
4     list_display = ('code', 'discount')
5     admin.site.register(Offer, OfferAdmin)

```

PRODUCTS

Offers

[+ Add](#) [✎ Change](#)

Change offer

HISTORY

Code:

Description:

Discount:

Let Products Show on the Website

views.py

```

1 from .models import Product
2
3 # /products -> index
4 # Uniform Resource Locator (Address)
5 def index(request):
6     return HttpResponse('Hello world')
7
8
9 def news(request):
10     products=Product.objects.all()
11     return HttpResponse('New Products')

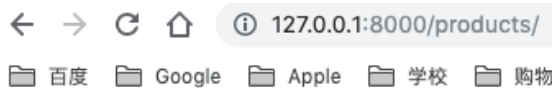
```

Create directory->Create HTML File



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

<http://127.0.0.1:8000/products/>



Products

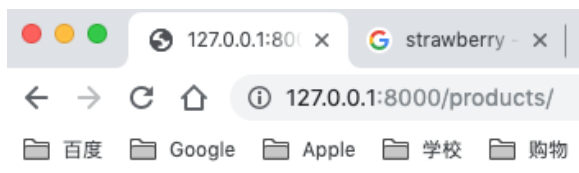
- Item 1
- Item 2
- Item 3

HTML file

```
1 <h1>Products</h1>
2 <ul>
3     {% for product in products %}
4     <!-- 使得可以执行Python语言-->
5         <li>{{product.name}}</li>
6     {% endfor %}
7 </ul>
```

views.py

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



Products

- Orange
- Strawberry

index.html

```

1 <h1>Products</h1>
2 <ul>
3     {% for product in products %}
4     <!-- 使得可以执行Python语言-->
5         <li>{{product.name}}({{product.price}})</li>
6     <!-- 显示名称, (价格) -->
7     {% endfor %}
8 </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

```

1 <!doctype html>
2 <html lang="en">
3     <head>
4         <!-- Required meta tags -->
5         <meta charset="utf-8">
6         <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-f
7
8         <!-- Bootstrap CSS -->
9         <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1
10
11         <title>Hello, world!</title>
12     </head>
13     <body>
14         {% block content %}
15         {% endblock %}
16
17     <!-- Optional JavaScript -->
18     <!-- jQuery first, then Popper.js, then Bootstrap JS -->
19     <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha3
20     <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper
21     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.m
22 </body>
23 </html>

```

index.html

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

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, i

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.boots

    <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.sl
    <script src="https://cdnjs.cloudflare.com/ajax/libs/
    <script src="https://stackpath.bootstrapcdn.com/boot
  </body>
</html>
```

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

{% block content %}
  <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>
{% endblock %}
```

样式变好看了

Products

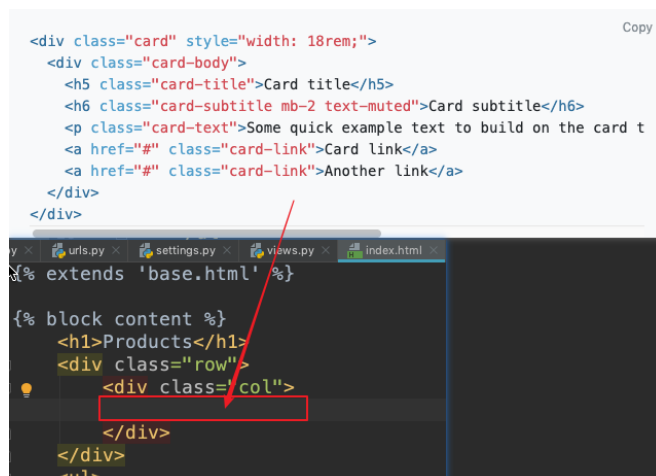
- Orange(\$1.99)
- Strawberry(\$2.99)

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

```
1 {% extends 'base.html' %}
2
3 {% block content %}
4     <h1>Products</h1>
5     <!--h1for heading-->
6     <div class="row">
7         {% for product in products %}
8         <!--      for loop-->
9         <div class="col">
10             <div class="card" style="width: 18rem;">
11                 <div class="card-body">
12                     <h5 class="card-title">Card title</h5>
13                     <h6 class="card-subtitle mb-2 text-muted">Card subtitle</h6>
14                     <p class="card-text">Some quick example text to build on the c
15                         card's content.</p>
16                     <a href="#" class="card-link">Card link</a>
17                     <a href="#" class="card-link">Another link</a>
18                 </div>
19             </div>
20         </div>
21         {% endfor %}
```

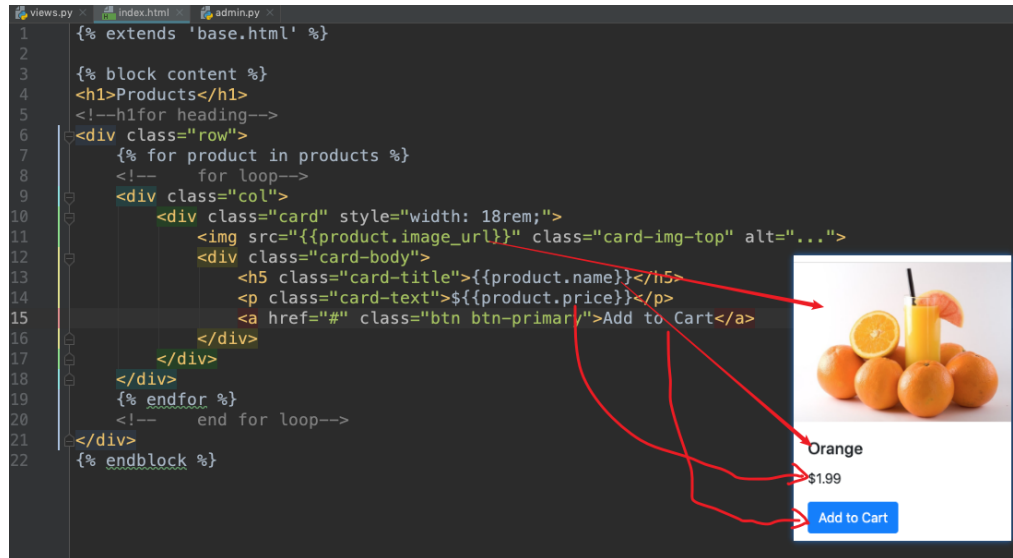


```

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

```

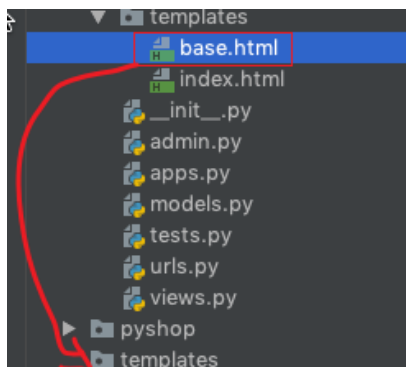
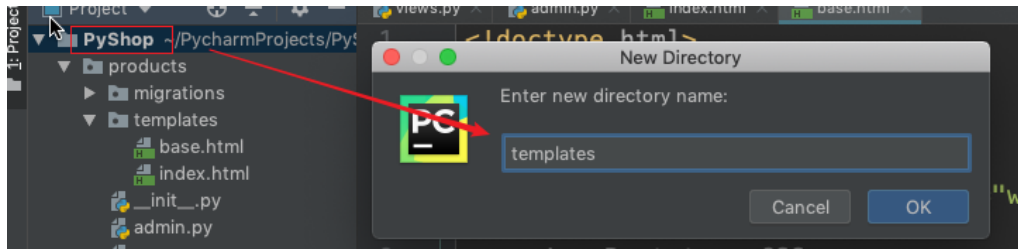
<!-- As a link -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
</nav>

```

Copy



Reuse it on multiple apps



移动后访问<http://127.0.0.1:8000/products/>

出现错误

Template-loader postmortem

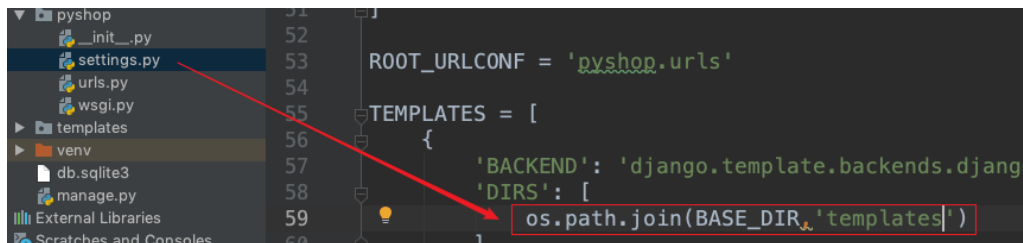
Pyshop

Django tried loading these templates, in this order:

Using engine django:

- `django.template.loaders.app_directories.Loader:`
`/Users/morris/PycharmProjects/PyShop/venv/lib/python3.7/site-packages/django/contrib/admin/templates/index.html` (Source does not exist)
- `django.template.loaders.app_directories.Loader:`
`/Users/morris/PycharmProjects/PyShop/venv/lib/python3.7/site-packages/django/contrib/auth/templates/index.html` (Source does not exist)
- `django.template.loaders.app_directories.Loader:`
`/Users/morris/PycharmProjects/PyShop/products/templates/index.html` (Source does not exist)

添加这一句



Pyshop

Products



Orange

\$1.99

Add to Cart

```
<div class="container">
  {% block content %}
  {% endblock %}
</div>
```

- __init__.py
- settings.py
- urls.py
- wsgi.py

templates

venv

db.sqlite3

manage.py

External Libraries

Scratches and Consoles

Terminal: Local × Local (2) ×

system check identi