所有的视频教程，都是给自己指明了一条学习的路径。让自己知道用什么程序什么包，怎么去用，而更加具体的方法，还必须要自己去阅读大量的文档说明，做大量的练习，进行深入的思考才能够掌握。掌握视频中的内容不是目的。

第零周：开始之前

## 课时1：付款第一件事：学习方式及加入班级

### 1. 讨论区的使用方法

* 第一步：课程讨论区>发起讨论>老师答疑区
* 第二步：描述问题

1. 选择课时
2. 一句话概括问题
3. 说明系统，版本
4. 给出报错截图，代码

### 2. 交作业

1. 作业方式：写作业学习笔记，发到微博@麻瓜编程
2. 哪些是作业：练习项目，实战作业都是作业
3. 怎么发作业学习笔记：打开简书：jianshu.com，写一篇文章。文章中包含成果，代码，总结。

使用markdown编辑文章能让文章样式很好看。

markdown基础用法，标题，代码，编号。

##小标题

'''

print(hello,world) #代码

'''

-第一件事

-第二件事

-第三件事

1. 分享到微博，微博加话题#python实战计划#，并@麻瓜编程

## 课时2：课程介绍：你能在四周获得什么？

只有动手做点东西出来，才能真正学会编程。

### 本课程特色

很多初学者很难找到适合学习的编程项目。文档和开源框架，那是给专业程序员看的，初学者看不懂。

于是有了这门课程，它和你看到的其他课程都不一样，有三个特别之处。

上手实践，而非啃理论。

第一，你将上手真实项目。通过大量的实战训练，循序渐进地提升。

生动易学，学编程不再枯燥。

第二，这是一门好看的编程课。有零基础学员说：这是我唯一能听懂的编程课，和看美剧似的，会上瘾。

你提出的每一个问题，都会得到解答。

第三，课程不仅仅是视频，还有专业程序员作为老师为你答疑与辅导。

### 有什么样的实战项目？

爬取电商网站，租房信息，美女图片集，二手商品信息。

第一周，我们从多个案例中掌握爬取网页的方法。 mongoDB

第二周，我们来学习数据库的使用，并爬取10W量级的数据。

第三周，我们对数据进行统计分析，可视化展示。发现数据的价值。

第四周，我们学习用Django框架搭建网站。并把数据分析成果展示出来。 动手完成Pinterest网站，二手行情网站。

### 科学的学习方法

在课程中，为初学者建立了一套科学的学习体系，把一个有难度的目标拆解成了3步：理解 > 模仿 > 实战。

第一步：理解视频学习。我们不会堆砌概念，而是通过一个个例子，让你更轻松的理解项目必要知识。

第二步：模仿，动手练习。正如学习母语，小孩子通过模仿成人说话而忘记。跟着每一节的视频，模仿着敲一遍代码，能加深印象。

第三步：实战，实战项目。学习编程，必须实战。通过大量的练习，才能训练你独立解决问题的能力，最终实现完整的爬虫系统项目。

第一周：学会爬取网页信息

第二周：学会爬取大规模数据

第三周：数据统计与分析

第四周：搭建Django网站

第四步会在每周循环一次，一共4周，完成挑战。

### 辅导老师能教我什么?

被问题卡住，却找不到答案是最难受的。加入班级群后，老师可以为你解答每一个困惑的问题。

除此之外，老师还会为你点评作业，专题辅导。

辅导老师就像是健身教练，帮助你更加有效的提升。

### 项目一小步，成长一大步

每一次作业的突破，都有满满的成就感，都是值得欣喜的成长。

不仅能学会课程中的项目，还能活学活用，独立完成更多项目。

他是一名大一学生，参加课程的目标是爬取豆瓣的讲座信息，为参加创业比赛的app做内容运营的储备。学习课程第一周，他就达到了目标。最终，想法变成了现实，参赛项目也获得了2000元的奖金。

拥有了亲手实现想法的能力，从此再也不求人。

他是一名产品经理，想为自己做一个时间管理的产品。在学习课程之前，还只停留在原型图阶段。而学习课程之后，终于拥有了新手创造的能力，你会慢慢了解，这种感觉有多快乐。

还可以解决生活中的难题，让生活更加高效。租房的问题。建一个网站。

为了解决租房的问题，学员爬取了所有的租房信息，从大数据中对比出合适的价格，从此不再被中介坑！懂得编程的人能比其他人节省更多的时间和精力，让编程去搞定那些琐碎重复的事情，让自己的生活更加高效。

什么投资项目能有4位收益？投资自己。

你还可以帮别人实现他们做不到的事情，从中获得成就感和收益。

什么投资项目能快速获得4倍收益，余额宝？股市？只有投资自己，才能快速获得翻倍的收益，能力的增值是最有价值的增值。

### 常见问题。

零基础可以参加吗？

必须可以。这门课程就是为初学者而设计的。零基础同学需要先阅读课程中的魔法教程，预习了python基础知识之后，再继续学习视频课程。

课程的形式是什么？我该怎么学习？

课程的形式是视频+项目练习+老师辅导。

开课之后，需要跟着班级的学习进度，每周抽时间完成这一周的课程内容，并且在每周六晚八点按时提交作业。老师会在开课后为大家提供作业点评和辅导。

四周之后，课程内容还能继续观看吗？

可以，四周是我们为大家安排的学习计划，而视频课程内容是可以永久观看的。这样在四周结束之后，你还可以重温。

讯飞资深后端工程师

从事全国最大的教育资源云的基础服务架构设计，擅长高并发基础服务架构设计与开发。熟悉JAVA、Python、C#、JS、C/C++等编程语言。Github地址是：<https://github.com/shenghe>

## 课时3：开始之前的准备

在我们开始之前，需要做好准备。

### 一、科学上网。

作为成年人，我们总是有选择的。那么就选择最好的产品和服务，节省我们宝贵的时间和精力去做更重要的事，而不是浪费在错误的选择上。

推荐使用云影 http://www.cloudss.net

科学上网并没有你想象的那么麻烦与复杂，现在已经有了许多非常好的产品解决方案。

http://next.36kr.com/posts/collections/75

### 二、电脑系统。

MAC, Windows和Linux系统都可以学习本课程进行Python开发。如果你想长期从事编程工作，建议把MAC或Linux作为开发环境。

### 三、python环境。

课程使用Pycharm作为编译器，推荐你也使用。原因是在编程时的自动提醒，安装环节都很方便。好的工具提升效率。

python环境安装和pycharm的安装方法都会在小白教程中说明。

### 四、基础知识。

### 五、学习方法。

实战计划是课程是按照这样的学习路径进行系统设计，这也是人类学习新技能的最佳学习路径。 理解 > 模仿 > 实战

第一步：理解。你需要做的是：看视频课程，理解并吸收知识点。

第二步：模仿。每节视频里面都会有课程安全，模仿着敲一遍代码，加深印象。

第三步：实战。通过练习题和大作业里面的项目大量练习。最终实现完整的爬虫系统项目。

我们通过这几个平台来学习：网易云课堂+QQ群。

在网易云课堂能看到视频及练习，讨论区会积累常见的问答。

初学者编程难免会遇到问题，这时可以在QQ群里找老师和同学讨论。

### 六、课程安排。

#### 第零周：准备。

1. 选择班级，加入相应QQ群。

2. 做好课前准备，课时3。

在Github能看到视频里面的源码，还有练习题答案。

https://github.com/mugglecoding/Plan-for-combating

麻瓜编程-问题归类和解答文档。

https://www.gitbook.com/book/ajkipper/mugglecoding-questions/details

https://ajkipper.gitbooks.io/mugglecoding-questions/content/

为了获得项目源码和参考答案，需要下载源码ZIP

https://pan.baidu.com/s/1i56NBdb

3. 零基础同学需预习基础知识。

#### 第一周：学会爬取网页

**1、完成本周课前准备**

一次性安装好所有的所需库、软件等，为后面开发环节节省时间。

**2、浏览课程内容**

照着敲一遍课程中的案例代码

在课时3-参考资源里面下载的压缩包里，带有[课程案例源码]的都是这节视频课程里面的项目代码。

**3、完成对应的练习项目**

挑战这一节的练习题，完成后可以对照参考答案。

在课时3-参考资源里面下载的压缩包里，带有[练习题答案]的都是这节这节练习题对应的参考答案。

**4、完成实战作业**

完成每节内容之后，开始挑战本周大作业

每一周的实战作业，需要在周六晚8点前提交你的作业到qq群文件，练习题作业不需要提交。

命名规则如下：

week1-没运行成功-林茜茜.py

week2-运行成功-林茜茜.py

week1表示是第几周大作业，如果是第二周就写week2

在本地的运行状态，有[运行成功][没运行成功]两种

你的QQ名

**5、实战作业讲解**

完成作业讲解

完成作业后，可以看实战作业讲解的视频。

在课时3-参考资源里面下载的压缩包里，进入类似[第X周大作业]的文件夹能看到大作业的参考答案。

#### 第二周：学会爬取大规模数据

每周课程都是这样的学习流程，按照章节目录的顺序，自己安排学习时间就可以。但要求每周六晚8点前提交本周的大作业。

#### 第三周：数据统计与分析

#### 第四周：搭建Django数据可视化网站

#### 四周之后

可以复习课程，视频可以永久观看，可以用所学知识，做更多类似项目......

第一周：学会爬取网页信息

## 课时5 第一周课前准备

### 课程介绍

实战计划课程手册——现在就开始创造

第一周：学会爬取网页信息

当你不知道怎么去爬一个网页的时候，并非是不懂python语法，而是因为是不了解网页是怎么构成的、网络协议是怎样的。所以，第一周课程将以一个全新视角看待爬虫，详解爬取网页的原理，并让你掌握爬取任何一个网页的方法与思路。

课程开始之前，先来安装好需要的软件和库。这周课程需要做这几项课前准备：

1. 学习使用Chrome浏览器查看网页源码

2. 学习使用pycharm创建网页文件

3. 学习安装库的方法

4. 安装库: lxml

5. 安装库: BeautifulSoup4

6. 安装库: Requests

别嫌麻烦哈，第三方库可是python的重要特性，安装好了库，我们就相当于站在巨人的肩膀上了，好多东西都不用自己去写了，直接调用库的方法就可以了。

### 1. 学习使用Chrome浏览器查看网页源码

安装Chrome

查看网页元素的代码

### 2. 学习使用pycharm创建网页文件

课程中的网页项目的文件是这样的结构: 包括images文件夹、css文件、html文件。html文件需要和其他两者放在同一文件夹中。

使用pycharm创建网页文件方法是这样的：

1) 刚进入pycharm时选择open，然后打开你下载好的网页项目的文件夹。如Plan-for-combating > week1 > 1\_1 > 1\_1answer\_of\_homework

2) 临时删除homework.html。自己新建一个html文件。

3) 看到这样的结构，左侧并列显示三项：html文件、css文件、images文件夹。这样就创建好了。

运行本地网页时记得关掉VPN，不然可能会无法显示。

### 3. 安装库的方法

这里先介绍安装库的通用方法，大多数库都可以使用这个方法来安装。但有少部分库是使用其他语言编译的，对这样的库后面会特别说明它的安装方法。

#### 在Mac上安装第三方库

**方法1：在pycharm中安装**

在pycharm的菜单中选择：File > Default Settings > 搜索project interpreter > 选择当前版本(Python环境) > 点击 + 添加库 > 输入库的名称 > 勾选install并点击Install Package

在安装成功后，pycharm会有成功提示。也可以在project interpreter这个界面中查看安装了哪些库，点 - 号就可以卸载不需要的库。

**方法2：在pip中安装**

安装好python环境之后，mac就可以直接支持pip，所以只需要输入一行：

pip3 install PackageName

#PackageName指的是你需要安装的某个库的名称，并不是一个具体的库的名字。

如果想安装到python2中，需要把pip3换成pip。

如果安装了python2和3g两个版本，可能会安装目录的问题，可以换成：

python3 -m pip install PackageName

如果想安装到python2中，需要把python3换成python

如果遇到权限问题，可以输入：

sudo pip install PackageName

安装成功后会提示：

Successfully installed PackageName

#### 在windows上安装第三方库

首先，确认你的windows电脑是多少位的。

然后，确认你的pip是否成功安装，在命令行cmd中输入

pip -- version

如果已经成功安装了pip，会提示类似下面的：

pip 9.0.1 from c:\python35\lib\site-packages (python 3.5)

如果你的电脑没有返回类似上面的提示，而是有了报错，提示你pip没有被定义，参考这个方法为你的Windows安装pip：

<https://taizilongxu.gitbooks.io/stackoverflow-about-python/content/8/README.html>

**方法一：在pycharm中安装**

在pycharm的菜单中选择：File > Default Settings > 搜索project interpreter > 选择当前版本(Python环境) > 点击 + 添加库 > 输入库的名称 > 勾选install并点击Install Package

在安装成功后，pycharm会有成功提示。也可以在project interpreter这个界面中查看安装了哪些库，点 - 号就可以卸载不需要的库。

**方法二：在pip中安装**

安装好python环境之后，mac就可以直接支持pip，所以只需要输入一行：

pip3 install PackageName

#PackageName指的是你需要安装的某个库的名称，并不是一个具体的库的名字。

如果想安装到python2中，需要把pip3换成pip。

如果安装了python2和3g两个版本，可能会安装目录的问题，可以换成：

python3 -m pip install PackageName

如果想安装到python2中，需要把python3换成python

如果遇到权限问题，可以输入：

sudo pip install PackageName

安装成功后会提示：

Successfully installed PackageName

**方法三：下载包安装**

有时候在windows上用前两种方法容易出问题，这时候你就手动安装吧。

进入pypi.python.org，搜索你要安装的库的名字，这时候有3种可能，

第一种是exe文件，这种最方便，下载满足你的电脑系统和python环境的对应的exe。

第二种是.whl类文件，好处在于可以自动安装依赖包。

第三中是源码，大都是zip、tar.zip、tar.bz2格式的压缩包，这个方法要求用户已经安装了这个包所依赖的其他包。例如pandas依赖于numpy。

如果你选择了下载.whl类文件，按如下方法进行安装

1. 到命令行执行

pip3 install wheel

等待执行完成，不能报错。

2. 从资源管理器中确认你下载的.whl类文件的路径，然后按shift+右键，选择在此处打开命令行。

然后在命令行输入:

pip3 install xxx.whl

如果下载了源码包，如下是安装方法

1. 解压包，进入解压好的文件夹，通常会看到一个setup.py的文件。在此处打开命令行，在命令行中输入：

python3 setup.py install

这个命令，就能把这个第三方库安装到系统里，即系统的python路径，默认是C:\Python3.5\Lib\site-packages

想卸载库的时候，找到python路径，进入site-packages文件夹，在里面删除库文件就可以了。

#### 在linux上安装第三方库

**方法1：使用系统的软件包管理来安装**

sudo apt-get install Python-PackageName

这样会安装到系统的默认python版本里，一般默认的是2。如果想更改默认的python版本，可以查看：

http://blog.csdn.net/eric\_sunah/article/details/8159745

指定ubuntu下的Python的运行版本

2012-11-07 21:52 22999人阅读 评论(12) 收藏 举报

分类： Python（25）

版权声明：本文为博主原创文章，未经博主允许不得转载。

问题描述：

Ubuntu 12中默认就自带了python的2.7版本，我自己安装了一个python3.2的版本，可是如果你不设置环境变量，那么在终端中运行的python都是系统默认的版本，这样就会导致3.2的程序在终端下无法运行，没有办法必须要将默认的python版本换到3.2.

问题分析：

其实在linux系统中多个python版本是可以共存的，只不过在终端中运行的时候，输入 python2.7 或者 python3.2 就能进入不同的版本了，而且在你的\*.py文件中可以用 ＃!/usr/bin/python 来指定程序的解释器版本，不过这样使用起来有些不方便，所以就把/usr/bin/python这个快捷方式的指向修改下好了。

解决方法：

sudo cp /usr/bin/python /usr/bin/python\_bak，先备份

sudo rm /usr/bin/python，删除

sudo ln -s /usr/bin/python3.2 /usr/bin/python，默认设置成python3.2

这样在终端中输入python默认就是 3.2版本了

**方法2：使用pip安装**

有时候使用前面的方法，安装的库可能不是最新的。或者你想做一个自由的少年，在python2和3之间任意切换，这时候可以使用pip这种方法。

**ubuntu安装pip的方法；**

**方法1：**

1.下载https://bootstrap.pypa.io/get-pip.py，把代码下到本地目录，然后用终端运行这个.py文件就行了。

2.代码放在ubuntu的用户主目录，然后使用sudo python get-pip.py安装pip，sudo python3 get-pip.py安装pip3

3.保持联网，脚本会自动下载一些东西，执行完成后，使用pip --version命令查看是否成功安装。

**方法2：**

1.安装easy\_install

sudo apt-get install python-setuptools

2.然后

sudo easy\_install pip

3.执行完成后，使用pip --version命令查看是否成功安装。

安装好了pip，就可以使用pip安装库了，输入：

pip3 install PackageName

如果想安装到python2，那么把上面的pip3换成pip就可以了。

**4、安装库：lxml**

**在MAC上安装lxml**

安装lxml之前需要先安装Command Line Tools，其中一种安装方法是，在终端中输入:

xcode-select --install

如果安装成功会提示Successful的字样。如果安装失败，可以使用brew或是下载 dmg的方式进行安装，具体方法请用Google搜索[Mac安装command line tools]

然后就可以安装lxml了，最快的安装方法是在终端中输入：

pip3 install lxml

关于pip的详细说明，还有其他安装方法，请查看前面的[3、学习安装库的方法]

**在Linux上安装lxml**

sudo apt-get install python3-lxml

也可以使用pip安装。请查看前面的[3、学习安装库的方法]

**在Windows上安装lxml**

1) 首先需要确认你的pip可以使用，在命令行中输入以下内容确认

pip --version

2) 下载lxml。

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml>

找到:

Lxml, a binding for the libxml2 and libxslt libraries.

lxml‑3.7.3‑cp27‑cp27m‑win32.whl

lxml‑3.7.3‑cp27‑cp27m‑win\_amd64.whl

lxml‑3.7.3‑cp34‑cp34m‑win32.whl

lxml‑3.7.3‑cp34‑cp34m‑win\_amd64.whl

lxml‑3.7.3‑cp35‑cp35m‑win32.whl

lxml‑3.7.3‑cp35‑cp35m‑win\_amd64.whl

lxml‑3.7.3‑cp36‑cp36m‑win32.whl

lxml‑3.7.3‑cp36‑cp36m‑win\_amd64.whl

需要根据自己的python版本和windoes版本进行选择。

3) 打开命令行，输入：

pip3 install wheel

需要等待执行完成，不能报错

4) 从资源管理器中确认你下载的.whl文件的路径，在此路径下打开命令行，输入：

pip install lxml‑3.7.3‑cp35‑cp35m‑win\_amd64.whl

**5、安装库：BeautifulSoup4**

无论是windows，mac还是linux，都可以使用pip来安装，在终端或命令行中输入：

pip3 install beautifulsoup4

也可以使用pycharm来安装，搜索beautifulsoup4就可以了。

**6、安装库：Requests**

Requests库的slogon是HTTP for humans，写给人类的HTTP。正如其名，使用体验确实是很友好。

Windows,mac和linux都可以使用pip来安装：

pip3 install requests

也可以使用pycharm来安装，搜索requests就可以了。

注：手册可能无法穷举所有问题，更多安装问题或使用说明可以查看官网或自行Google:

lxml:

http://lxml.de/installation.html

BeautifulSoup中文文档：

<http://www.crummy.com/software/BeautifulSoup/bs4/doc/index.zh.html>

Requests中文文档：

<http://docs.python-requests.org/zh_CN/latest/>

## 课时6 第一周课前准备视频：python安装及环境配置

### windows下python环境搭建

直接下载官方网站exe安装程序，安装时选择添加到环境变量。

将安装目录所在路径配置到环境变量

### linux下python环境搭建

sudo apt-get install python3.5

源码安装方式：

tar -zxvf python-xxx.tgz

cd python-xxx

./configure #检查当前配置是否正确

make #编译源码

sudo make install

### mac下python环境搭建

直接双击安装python pkg文件。

## 课时7 第一周课前准备视频：python库安装及使用。

pip和easy\_install安装方式

### windows下python库安装方法

cmd

pip --version

python35 > Scripts easy\_install.exe easy\_install-3.5.exe pip.exe pip3.5.exe

修改环境变量，把python35/Scripts路径加到path中去。

重 新启动命令行

可以使用pip或easy\_install来安装库

pip

pip install bs4

easy\_install requests

卸载

pip uninstall bs4

### Linux下python库的安装方法

pip安装：

sudo apt-get install python3-pip

或者

wget <https://bootstrap.pypa.io/get-pip.py>

python3 get-pip.py

pip3 install bs4

easy\_install安装

wget <https://bootstrap.pypa.io/ez_setup.py>

python3 ez\_setup.py

### Mac OS

pip安装

curl <https://bootstrap.pypa.io/get-pip.py>

python3 get-pip.py

easy\_install安装

curl -O <https://bootstrap.pypa.io/ez_setup.py>

python3 ez\_setup.py

pip一键安装库

安装好了pip之后，可以直接输入如下命令安装所需类库

pip install requests bs4 pymongo mongoengine jupyter Django charts

pip3 install requests bs4 pymongo mongoengine jupyter Django charts

pycharm安装库方法

settings > Project Interpreter > choose python35 or python27 > click + to add package > requests >install package > requests aliyun > Django

## 课时8 第一节课程：认识网页的构成

网页类似我们住的房子，Html是结构，如三室一厅一卫一厨，区别网页中每一块内容的功能，css是样式，相当于房子的装修。JavaScript是功能，类似于房间中的电器。

<div> </div> 代表网页中的一个区域，在这个区域可以加入内容。

<p> </p> 分段

<li> </li> 列表

<img> 图片

<h1> </h1> 标题

<a href = “”> 链接

使用div标签来区分网页的不同区域或不同的功能

学习网页知识最好的方法莫过于自己动手去做一个网站。

实战计划生存法则

1、编程是一个Learning by doing的技能，当你能够完成这道练习题，你才真正学会了这节课的知识点；

2、如果做练习的过程中遇到困难，善用搜索找答案；

3、初学者都不爱看文档，可是文档很重要。所以我们的一些题目会引导你去看文档，当做字典去查阅就好；

4、虽然我们提供了练习题参考答案，但你最好在独立完成之后，再对照答案；

5、当正式参加四周课程后，需要每周提交实战作业，并从老师的作业点评中提升自己。

## 课时9 第一节练习项目：动手做自己的网页

## 课时10 第一节练习项目详解：动手做自己的网页

## 课时11 第二节课程：解析网页中的元素

学会筛选出所有评分大于3的文章，展现标题和对应的分类

第一步：使用BeautifulSoup解析网页

soup = BeautifulSoup(html, “lxml”) lxml为解析网页所用的库。

soup=汤，html=做汤的原材料，lxml=食谱。

解析网页所用的库一般有5种：

html.parser: BeautifulSoup内置引擎，无需安装

lxml HTML

lxml XML: pip install lxml

html5lib：pip install html5lib

1. 使用BeautifulSoup解析网页

第一种方式：

fs = open(“文件地址”,”r”)

print(fs.read())

fs.close() #必须要调用close()函数关闭文件，否则会内存泄漏。

第二种方式：

with open(“文件地址”,”r”) as fs:

print(fs.read())

**from** bs4 **import** BeautifulSoup  
  
**with** open(**"D:/David/Desktop/fourweeks/week1/02 class/web/new\_index.html"**) **as** wb\_data**:** soup **=** BeautifulSoup(wb\_data, **'lxml'**)  
 print(soup.prettify())

第二步：描述要爬取的信息在哪里

**两种不同的选取方式**

BeautifulSoup只能识别css selector的方式

点击网页中的Sardinia’s top 10 beaches中的图片元素，检查，在对应的代码上右键，copy selector，copy Xpath

CSS Selector:

按照网页中元素的样式进行选择。

谁，在哪，第几个，长什么样子。

body > div.main-content > ul >li:nth-child(1) >img

XPath: 用父节点，子节点，子孙节点，兄弟节点的方式进行元素的区分。谁，在哪，第几个。

/html/body/div[2]/ul/li[1]/img

2. 获取单个元素的地址

**image:**body > div.main-content > ul > li: nth-child(1) > img 原来的css selector  
**title:**body > div.main-content > ul > li:nth-child(1) > div.article-info > h3 > a  
**category:**body > div.main-content > ul > li:nth-child(1) > div.article-info > p.meta-info > span:nth-child(1)

**description:**body > div.main-content > ul > li:nth-child(1) > div.article-info > p.description

**rate:**body > div.main-content > ul > li:nth-child(2) > div.rate > span

由于Beautifulsoup不支持nth-child语法，要把上面的所有nth-child(1)改为nth-of-type(1)

这样就能选择出所有的第一个元素。

3. 获取多个元素的地址，并进行测试

要爬取到所有的元素，多复制两个相似的元素就能找到规律

body > div.main-content > ul > li > img 有很多images，去掉具体的位置信息，就能得到所有的图片。同样得到其它元素的地址：

**images:**body > div.main-content > ul > li > img   
**titles:**body > div.main-content > ul > li > div.article-info > h3 > a  
**categories:**body > div.main-content > ul > li > div.article-info > p.meta-info > span

**descriptions:**body > div.main-content > ul > li > div.article-info > p.description

**rates:**body > div.main-content > ul > li > div.rate > span

代码测试：

**from** bs4 **import** BeautifulSoup  
  
**with** open(**"D:/David/Desktop/fourweeks/week1/02 class/web/new\_index.html"**) **as** wb\_data**:** soup **=** BeautifulSoup(wb\_data, **'lxml'**)  
titles = soup.select(**"body > div.main-content > ul > li > div.article-info > h3 > a"**)  
images = soup.select(**"body > div.main-content > ul > li > img"**)  
categories = soup.select(**"body > div.main-content > ul > li > div.article-info > p.meta-info"**)  
descriptions = soup.select(**"body > div.main-content > ul > li > div.article-info > p.description"**)  
rates = soup.select(**"body > div.main-content > ul > li > div.rate > span"**)  
  
print(images,titles,cates,descs,rates,sep**="\n==============\n"**)

或者：

**for** title,image,categorie,description,rate **in** zip(titles,images,categories,descriptions,rates):  
 print(title,image,categorie,description,rate)

第三步：筛选所需要的信息。从标签中获得你要的信息，并把信息放在一定的数据容器中，方便我们的查询。

4. 先提取出一个元素的信息

**from** bs4 **import** BeautifulSoup  
  
**with** open(**"D:/David/Desktop/fourweeks/week1/02 class/web/new\_index.html"**) **as** wb\_data**:** soup **=** BeautifulSoup(wb\_data, **'lxml'**)  
titles = soup.select(**"body > div.main-content > ul > li > div.article-info > h3 > a"**)  
images = soup.select(**"body > div.main-content > ul > li > img"**)  
categories = soup.select(**"body > div.main-content > ul > li > div.article-info > p.meta-info >span"**)  
descriptions = soup.select(**"body > div.main-content > ul > li > div.article-info > p.description"**)  
rates = soup.select(**"body > div.main-content > ul > li > div.rate > span"**)

**for** title **in** titles**:** print(title.get\_text())

这样就把标题的信息都提取出来了。

要把所有的信息都提取并保存起来，建立一个字典。

5. 提取所有元素的信息，放在一个字典中。

**from** bs4 **import** BeautifulSoup  
  
**with** open(**"D:/David/Desktop/fourweeks/week1/02 class/web/new\_index.html"**) **as** wb\_data**:** soup **=** BeautifulSoup(wb\_data, **'lxml'**)  
titles = soup.select(**"body > div.main-content > ul > li > div.article-info > h3 > a"**)  
images = soup.select(**"body > div.main-content > ul > li > img"**)  
categories = soup.select(**"body > div.main-content > ul > li > div.article-info > p.meta-info >span"**)  
descriptions = soup.select(**"body > div.main-content > ul > li > div.article-info > p.description"**)  
rates = soup.select(**"body > div.main-content > ul > li > div.rate > span"**)

**for** title,image,description,rate,category **in** zip(titles,images,descriptions,rates,categories):  
 data = {  
 **"title"**:title.get\_text(),  
 **"image"**:image.get(**"src"**),  
 **"category"**: category.get\_text(),  
 **"description"**: description.get\_text(),  
 **"rate"**:rate.get\_text(),

*#获取图片的链接，图片的链接不是文本，而是标签的属性，使用get方法来获得标签的属性。* **"image":**image.get(**"src"**),  
 }  
 print(data)

6. category爬取的信息不完整

运行发现所有的category属性都只取了第一个，要在它的父级标签进行取值

**from** bs4 **import** BeautifulSoup  
  
**with** open(**"D:/David/Desktop/fourweeks/week1/02 class/web/new\_index.html"**) **as** wb\_data**:** soup **=** BeautifulSoup(wb\_data, **'lxml'**)  
categories = soup.select(**"body > div.main-content > ul > li > div.article-info > p.meta-info"**)  
print(categories)

cates及它的父级标签就被筛选出来了

7. category信息的处理

**from** bs4 **import** BeautifulSoup  
**with** open(**"D:/David/Desktop/fourweeks\_code/week1/02\_class/web/new\_index.html"**) **as** wb\_data:  
 soup = BeautifulSoup(wb\_data, **'lxml'**)  
titles = soup.select(**"body > div.main-content > ul > li > div.article-info > h3 > a"**)  
images = soup.select(**"body > div.main-content > ul > li > img"**)  
categories = soup.select(**"body > div.main-content > ul > li > div.article-info > p.meta-info"**)  
descriptions = soup.select(**"body > div.main-content > ul > li > div.article-info > p.description"**)  
rates = soup.select(**"body > div.main-content > ul > li > div.rate > span"**)  
  
**for** title,image,description,rate,category **in** zip(titles,images,descriptions,rates,categories):  
 data = {  
 **"title"**:title.get\_text(),  
 **"image"**:image.get(**"src"**),

#构造category的数据结构时要进一步处理，要使用stripped\_strings方法，能够获得所有父级标签下的文本信息，可以理解为get\_text处理多个文本的高级版本。由于获得的标签是成组的，还要进行列表化。  
 **"category"**: list(category.stripped\_strings),  
 **"description"**: description.get\_text(),  
 **"rate"**:rate.get\_text(),  
 }  
print(data)

8. 把所有结果放在一个列表中，对结果进行初步的分析，找出所有评分大于3的信息。

**from** bs4 **import** BeautifulSoup  
info = []  
**with** open(**"D:/David/Desktop/fourweeks\_code/week1/02\_class/web/new\_index.html"**) **as** wb\_data:  
 soup = BeautifulSoup(wb\_data, **'lxml'**)  
titles = soup.select(**"body > div.main-content > ul > li > div.article-info > h3 > a"**)  
images = soup.select(**"body > div.main-content > ul > li > img"**)  
categories = soup.select(**"body > div.main-content > ul > li > div.article-info > p.meta-info"**)  
descriptions = soup.select(**"body > div.main-content > ul > li > div.article-info > p.description"**)  
rates = soup.select(**"body > div.main-content > ul > li > div.rate > span"**)  
  
**for** title,image,description,rate,category **in** zip(titles,images,descriptions,rates,categories):  
 data = {  
 **"title"**:title.get\_text(),  
 **"image"**:image.get(**"src"**),  
 **"category"**: list(category.stripped\_strings),  
 **"description"**: description.get\_text(),  
 **"rate"**:rate.get\_text(),  
 }  
 info.append(data)  
*#打印测试  
# for i in info:  
# print(i)***for** i **in** info:  
 **if** float(i[**"rate"**]) > 3:  
 print(i[**"title"**],i[**"cate"**])

## 课时12 第二节练习项目：爬取商品信息

## 课时13 第二节练习项目详解：爬取商品信息

css属性选择器详解

<http://www.w3school.com.cn/css/css_selector_attribute.asp>

zip用法详解

<https://docs.python.org/2/library/functions.html#zip>

Homework

解析本地购物网页,获取以下信息:

图片地址，价格，商品标题，评分量，评分星级。

难点：如何统计有几个\*

查阅BeautifulSoup的中文文档，阅读其中关于find\_all()的内容，从中发现如何使用find\_all统计星的个数。

<https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/#find-all>

title:

body > div:nth-child(2) > div > div.col-md-9 > div:nth-child(2) > div:nth-child(1) > div > div.caption > h4:nth-child(2) > a

description:

body > div:nth-child(2) > div > div.col-md-9 > div:nth-child(2) > div:nth-child(1) > div > div.caption > p

image:

body > div:nth-child(2) > div > div.col-md-9 > div:nth-child(2) > div:nth-child(1) > div > img

review:

body > div:nth-child(2) > div > div.col-md-9 > div:nth-child(2) > div:nth-child(1) > div > div.ratings > p.pull-right

star:

body > div:nth-child(2) > div > div.col-md-9 > div:nth-child(2) > div:nth-child(1) > div > div.ratings > p:nth-child(2) > span:nth-child(1)

body > div:nth-child(2) > div > div.col-md-9 > div:nth-child(2) > div:nth-child(1) > div > div.ratings > p:nth-child(2) > span:nth-child(2)

price:

body > div:nth-child(2) > div > div.col-md-9 > div:nth-child(2) > div:nth-child(2) > div > div.caption > h4.pull-right

得到所有元素的css地址

title:

body > div > div > div.col-md-9 > div > div > div > div.caption > h4 > a

description:

body > div > div > div.col-md-9 > div > div > div > div.caption > p

image:

body > div > div > div.col-md-9 > div > div > div > img

review:

body > div > div > div.col-md-9 > div > div > div > div.ratings > p.pull-right

star:

body > div > div > div.col-md-9 > div > div > div > div.ratings > p > span

price:

body > div > div > div.col-md-9 > div > div > div > div.caption > h4.pull-right

使用soup.select得到的列表

观察发现，标题信息，浏览量，价格是放在标签中间的，可以利用get\_text()函数得到中间文本的内容

title\_content = title.get\_text()

review\_content = review.get\_text()

price\_content = price.get\_text()

图片的链接放在了<img>标签的src属性中，可以利用get()函数得到属性文本内容：

image\_content = image.get(“src”)

星星的数目就是soup.select()匹配的标签数目，所以星星的数目为：

star\_count = len(star)

## 课时14 第三节课程：真实世界中的网页解析

用Requests + BeautfulSoup爬取TripAdvisor

第1步：了解服务器与本地的交换机制，选择正确的方法才能正确获取信息。

Request方法 Http1.1

get post head put options connect trace delete

get和post是最常用的两种方法。

点击按钮时get

发微博时post

爬虫模拟的就是这两种方法。

其实，知道了get的方法，90%的网页数据都能够爬取下来。

使用get的方法至少要包含如下基本信息：

GET /page\_one.html HTTP/1.1

Host: [www.sample.com](http://www.sample.com)

第2步：了解解析真实网页的方法和思路

## 课时15 第三节练习项目：爬取租房信息

Level 1

进入小猪短租，从列表页点击进入第一个详情页

<http://bj.xiaozhu.com/search-duanzufang-p1-0>

在详情中爬取以下信息：

标题

地址

日租金

第一张房源图片链接

房东图片链接

房东性别

房东名字

小提示：

难点：房东性别

这不是一个可以直接爬取的信息，而是需要你来做一些处理。只要运用if else这个条件判断就能解决了。

Level 2

爬取300个房源，还是刚才的那些信息。

小提示：

抓取1个详情页容易，但需要抓取300个详情页的时候，怎么办呢？

我们需要先从列表页获取每个详情页的链接，构建一个函数帮我们做这件事，方法有很多种，这就是在考验你的基础知识够不够扎实了。



## 课时16 第三节练习项目详解：爬取租房信息

从商品列表页得到所有的商品详情页链接，然后从商品详情页提取商品信息

新建 main.py

打开main.py，写好基本信息，定义两个函数，一个获取列表页的所有链接，一个获取详情页面信息

#获取详细信息页面

def get\_links():

pass

def get\_detail\_info():

pass

分析列表页链接规律

打开<http://sh.xiaozhu.com>,点击 第2页，第3页，观察浏览器地址栏地址的规律

[http://sh.xiaozhu.com/search-duanzufang-p{}-0/](http://sh.xiaozhu.com/search-duanzufang-p%7b%7d-0/)

替换括号所在位置为页码对应数字，数字为几则是第几页

生成10个列表页面链接

列表= [变量 for 变量 in range(开始数字,结束数字,数字间隔)]

从列表页面获取详情页面的链接

难点是找到详情页地址所在标签，还需要注意，得到css path后，要删除:nth-child(),否则会报错

请求详情页，获取返回的数据

写解析列表页面的函数，先requests库请求得到数据，然后通过BeautifulSoup库解析。

def get\_detail\_info(url):

wb\_data = requests.get(url)

#这里采用lxml引擎解析请求得到的列表页面数据

soup = BeautifulSoup(wb\_data.text,’lxml’)

根据css path，提取相应信息列表

chrome浏览器打开网页，把鼠标放在相应的信息上，审查元素，copy css paty，去年：nth-child()

循环标签列表，得到标签的信息

for 得到的是标签，可通过get\_text()函数提取信息，get\_text()得到文本，get()得到属性内容。

不同的性别，class属性内容是不同的，需要单独写函数做判断。

继续写通过class内容判断属性的函数

for得到的是标签，可通过get\_text()和get()函数提取信息，get\_text()得到文本，get()得到属性内容。

注意：sex.get(“class”)得到的是列表，不是字符串。

<http://sh.xiaozhu.com/>

<http://sh.xiaozhu.com/search-duanzufang-p2-0/>

<http://sh.xiaozhu.com/search-duanzufang-p3-0/>

#page\_list > ul > li:nth-child(1) > a

#page\_list > ul > li:nth-child(2) > a

href属性即是对应的地址

#page\_list > ul > li:nth-child(1) > a > img

<img class="lodgeunitpic" title="经典石库门\*私人影院\*5分钟地铁\*行李寄存" src="http://image.xiaozhustatic1.com/12/8,0,99,5441,1799,1200,3ba2d876.jpg" lazy\_src="finish" alt="经典石库门\*私人影院\*5分钟地铁\*行李寄存" style="height: 214px;">

<a href="/Attraction\_Review-g308272-d311595-Reviews-The\_Bund\_Wai\_Tan-Shanghai.html" onclick="ta.setEvtCookie('Attraction\_List\_Click', 'POI\_click', 'name', 1, '/Attraction\_Review')" target="\_blank">上海外滩</a>

**"title"**: title.get\_text(),

<img alt="上海外滩" width="180" style="height: 180px; width: 239px; margin-left: -29px;" id="lazyload\_1860739714\_3" class="photo\_image" height="180" src="https://media-cdn.tripadvisor.com/media/photo-f/0a/cc/fc/09/photo0jpg.jpg">  
**"image"**: image.get(**"src"**),

<div class="p13n\_reasoning\_v2">

<a onclick="ta.setEvtCookie('Attraction\_List\_Click', 'POI\_click', 'tags', 1, '/Attractions')" href="/Attractions-g308272-Activities-c47-t19-Shanghai.html"><span class="matchedTag noTagImg">史迹徒步游区域</span></a>

<a onclick="ta.setEvtCookie('Attraction\_List\_Click', 'POI\_click', 'tags', 2, '/Attractions')" href="/Attractions-g308272-Activities-c47-t163-Shanghai.html"><span class="matchedTag noTagImg">景点与地标</span></a>

</div>  
**"category"**: list(category.stripped\_strings),

<a target="\_blank" href="/Attraction\_Review-g308272-d311595-Reviews-The\_Bund\_Wai\_Tan-Shanghai.html#REVIEWS">

12,665条点评

</a>  
**"commentNum"**: int(commentNum.get\_text().replace(**'\n'**, **''**).replace(**','**, **''**).replace(**'条点评'**, **''**)),

<img class="sprite-ratings" src="https://static.tacdn.com/img2/x.gif" alt="4.5 分，共 5 分">  
**"rating"**: float(rating.attrs[**"alt"**].replace(**" 分，共 5 分"**, **""**)),

获取属性值 image.get(“src”)

获取标签内容 title.get\_text()

## 课时17 第四节课程：如何获取网页中的动态数据。

使用爬虫抓取网站异步加载数据

第1步：什么是异步加载？

有些网页，不需要点击换页在一页中就能持续的加载内容。

<https://knewone.com/>

新浪微博评论

豆瓣电影

第2步：如何抓取异步加载数据？

network > XHR > 刷新

点中某个页面，在headers部分，就能找到它对应的网址

异步加载数据是由JS来控制的

1. 向网址请求，并解析网页

2. 描述要爬取的元素位置

3. 筛选所需的信息

4. 构造函数，爬取连续多页信息

每一步都要进行测试

这节课我们爬取了一个异步加载的网站，KnewOne

理解了异步加载的原理

学会了如何在Request中获取URL，从而爬取异步加载网页。

祝贺你已经学会了爬取这个世界上的绝大多数网页了。

挑战大作业

爬取58同城二手商品一页列表页中的商品详情信息

## 课时18 第四节练习项目：爬取MM图片

<http://weheartit.com/inspirations/taylorswift>

爬取前20页，获取图片并下载到本地

小提示：

1。你会看到Request url是类似这样的：

<http://weheartit.com/inspirations/vsfashionshow?scrolling=true&page=2&before=209640847>

后面的before参数不用管，不影响输出，爬page参数就可以了。

2。如何把图片下载到本地：

可以参考煎蛋的爬虫代码。

<http://mp.weixin.qq.com/s?__biz=MzAwOTcyNDE1Mw==&mid=401003733&idx=1&sn=4c42d66f9beb27158eab61837eb421c3#rd>

<http://jandan.net/ooxx/page-2410>

## 课时19 第四节练习项目详解：爬取MM图片

网络代理介绍与使用

网络代理，中转请求，用于绕过防火墙的网络限制或网站的ip限制，一般IP:端口的形式

proxies = {‘http’: ‘127.0.0.1:8888’}

http:代理的协议类型

127.0.0.1:8888:网络代理的地址

如何查找网络代理

搜索引擎搜索“免费网络代理”，不稳定且有效时间短

淘宝搜索

预备知识相关文档：

科学上网工具：

<http://mp.weixin.qq.com/s?__biz=MzAwOTcyNDE1Mw==&mid=2656578226&idx=1&sn=9674cbb38fb73355e0427a3d5060348f&scene=0#wechat_redirect>

免费网络代理：

<http://www.xicidaili.com/>

写好代码的大致结构

打开创建的main.py文件，写好基本信息，定义两个函数，一个运行入口，一个下载图片

from bs4 import BeautifulSoup

import requests

def download(url):

pass

def main():

pass

#只有直接运行脚本才会执行main函数，外部调用不会执行main()函数。

if \_\_name\_\_ = ‘\_\_main\_\_’:

main()

完善主函数，获取1-10页所有图片链接

继续在main函数中补充代码，找到图片列表页面的规律，发现只要替换链接参数即能进入对应页面

url = ‘http://weheartit.com/inspirations/beach?page={}’.format(page)

def main()

#获取1-10页的图片

for page in range(1,10):

#找规律，发现只有替换请求链接的page参数即可进入相应的页面

url = ‘http://weheartit.com/inspirations/beach?page={}’.format(page)

完善主函数，解析得到的数据，提取图片地址

请求得到的url，伪装爬虫为浏览器，加上header信息，为了避免网络限制，使用代理绕过网络封锁。

def main()

#获取1-10页的图片

for page in range(1,10):

#找规律，发现只有替换请求链接的page参数即可进入相应的页面

url = ‘http://weheartit.com/inspirations/beach?page={}’.format(page)

r = requests.get(rul, proxies=proxies, headers = headers)

#检查是否正常访问，异常访问返回的状态码不是200，异常就跳过

if r.status\_code != 200:

continue

#提取页面的图片，得到地址，然后下载

soup = BeautifulSoup(r.text, ‘html.parser’)

#得到的是图片标签列表，css selector通过审查元素得到

imgs = soup.selcet(‘img.entry\_thumbnail’)

for img in imgs:

src = img.get(‘src’)

download(src)

完善下载函数，请求图片地址

请求图片地址，检查图片地址文件是否存在，存在的话分析url得到图片名称，生成保存地址，然后保存

def downlaod(url):

r = requests.get(url, proxies = proxies, headers = headers)

if r.status\_code !=200:

return

#http://www.example.com/test.pgn?a=1

#第一个split，获取问题前面的字符

#第二个split，获取倒数第二个一个斜杠后面的内容，因为最后一个斜杠后面的字符串都一样，不能用

file\_name = url.split(‘?’)[0].split(‘/’)[-2]

target = ‘./{}.jpg’.format(file\_name)

with open(target, ‘wb’) as fs:

fs.write(r.content)

print(‘%s ==> %s’%(url, target))

运行，解决可能的问题

执行python main.py运行脚本，如果出错，一一解决，可能的错误原因如下：

1。代理失效，重新换代理

2。输出为空，可能网站已经做了修改，css selector已经找不到图片的地址，需要修改。

3。连接网站超时，可能代理已经被网站封锁，更改代理地址。

## 课时20 第一周实战作业，爬取一页商品数据

Level 1

<http://bj.58.com/pbdn/0/>

进入58这个页面，爬取列表页中除了转转，推广商品之外的正常商品

在详情页中爬取这些信息

类目

标题

发帖时间

价格

成色

区域

小提示：

如果爬取频率过于频繁，58可能会反爬取。所以，运用你在这周所学的所有知识，躲开反爬取吧。

Level 2

增加一个字段，爬取浏览量信息

小提示：

查看网页的源代码，发现浏览器的值为0，而审查元素看到的浏览器量不为空，说明浏览量这个数值是通过js获得的。

那么，就有很多方法可以搞定了。

比如说可以点击netword刷新网页，观察哪个url的response返回的值包含浏览量。

找到之后，接下来就是拼接url解析数据了。

经过练习，你会发现爬取网页的要点，最重要的是描述元素位置

学习的捷径：多练习，多尝试，不懂的地方查阅文档

还学会了从列表页中获取详情页链接的方法

还搞明白了怎么抓取js中的浏览量

## 课时22 第一周补充资料

谁需要读补充资料？

补充资料提供给学有余力的同学，在掌握了课程内容之后，还想了解更多原理、更多延伸内容的时候，可以来阅读这些资料。

对于基础稍微薄弱些的同学，也不要作为负担，这些不是必修课，如果看不下去，不要强迫自己，先放在这里，总有一天你能看懂这些资料。

补充资料怎么读？

一个核心原则，在你需要的时候阅读。这些资料大多像字典一样，没事翻字典是读不下去的，但当你需要了解某个细节的时候，这些资料能给你最精准的解读。

我们将以链接形式给出资料，一方面是尊重原作者的创作，另一方面也保持了资料的原汁原味和即时更新。

为什么补充资料大多是英文的？

编程语言本身是以英文为基础的，好的内容也大多出身于国外。而不好的翻译会让你误解原谅的本意。所以，如果要成为一个优秀的开发者，从试卷读懂英文开始吧。

以后我怎样自己找到这些资料？

你可以留心观察我们给出的资料的名字，比如带有cheatsheet，cookbook等等，下次你在搜索的时候加上这些词，就能找到类似的资料。

html5 new tag cheat sheet(en)

<http://websitesetup.org/HTML5-cheat-sheet.pdf>

解读：这里列举了所有的html5标签，这里并不是要你成为一个卓越的前端工程师，而是在遇到陌生的标签的时候，记得到这里来查阅。

BeaufitulSoup官方文档(cn)

<http://beautifulsoup.readthedocs.io/zh_CN/latest/>

解读：如果你认真做了1,2练习题，那么应该已经读过这份文档的一小部分了，和那次一样，在需要时查阅就好。

xpath,css的语法说明(en)

<http://www.cheat-sheets.org/saved-copy/Locators_groups_1_0_2.pdf>

<http://www.cheat-sheets.org/saved-copy/Locators_table_1_0_2.pdf>

解读：这里是xpath,css的语法大全，看了之后，你就会发现，课程中总结的用法是爬取网页的最少必要知识——让你最短时间内学到最关键的知识。更多的高级用法，你可以对照资料学习。

Requests官方文档(cn)

<http://docs.python-requests.org/zh_CN/latest/user/quickstart.html>

解读：Requests库的标语是HTTP for Humans，为人类设计的http库。这份文档简明易读，包括二进制、json等等高级用法都有讲解。

超文本传输协议——http1.1(cn)

<http://files.blogjava.net/sunchaojin/http1.3.pdf>

解读：在现实生活中我们需要遵循法律，而http协议就是网络环境中大家共同遵循的规则。这份文档为了追求精确严谨，可能会让普通人觉得很晦涩难懂。这样对比，你就更容易发现，我们的课程是如何把原本复杂的东西精简到你能理解的程度了。

第二周：爬取大规模数据

## 课时23 第二周课前准备

在上一周的基础上，我们继续爬取58同城上的30万数据，并把数据存放在炫酷的MongoDB中，开始做爬虫的主人吧。

课前准备

为了使用MongoDB，我们需要先安装好这三项：

1. 安装MongoDB

2. 安装第三方库Pymongo

3. 安装pycharm的MongoDB插件

## 课时24 MongoDB安装及环境配置

**Windows安装MongoDB**

**1. windows安装相关配置**

查看系统版本和多少位

wmic os get caption

wmic os get osarchitecture

**2. 下载，安装，修改安装路径**

**3. 把bin路径添加到系统环境变量中**

**4. 手动设置MongoDB环境**

新建md C:\MongoDB\Server\3.4\data\db

cmd进入bin目录，以管理员身份运行，dir相当于ls

mongod --dbpath C:\MongoDB\Server\3.4\data\db

如果数据目录中包含了空格，要用双引号括起来，如：

mongod --dbpath C:\MongoDB\Server\3.4\”mongo db data”\db

**5. 启动MongoDB**

在命令行输入

mongod

如果出现waiting for connections on port 27017就说明启动成功了。

打开浏览器，输入localhost:27017

如果无法启动服务，删除 db/mongod.lock文件，bin文件夹中运行mongod --repair

再次运行

mongod --dbpath C:\MongoDB\Server\3.4\data\db

浏览器中查看能否连接

如果出现MongoDB服务启动时报"Windows 无法启动Mongo DB服务 错误:1067 进程意外终止"

MongoDB安装目录\data\将此文件夹下的mongod.lock删除

mongod.exe --config E:\ruanjian\MongoDB\mongod.cfg --remove

mongod.exe --config E:\ruanjian\MongoDB\mongod.cfg --install

net start mongdb

**6) 连接MongoDB**

不要关闭刚才的命令行窗口，再打开一个命令行窗口，输入

mongo

如果出现类似以下提示，说明服务成功启动

MongoDB shell version v3.4.2

connecting to: mongodb://127.0.0.1:27017

MongoDB server version: 3.4.2

**7) 把mongodb添加到系统服务中，mongodb就能随计算机自动启动服务了。**

**关闭之前打开的所有mongodb的命令行**

新建data/log/mongod.log

新建/mongoDB/3.4/mongod.cfg, 指定相关配置路径

systemLog:

destination: file

path: C:\MongoDB\Server\3.4\data\log\mongod.log

storage:

dbPath: C:\MongoDB\Server\3.4\data\db

接下来我们将mongodb放到windows的服务中

重新打开一个命令行，必须以管理员权限打开

输入命令

mongod.exe --logpath d:/develop/MongoDB/data/logs/mongodb.log --logappend --dbpath d:/develop/MongoDB/data/db --directoryperdb --serviceName MongoDB -install

mongod.exe --config <cfg文件路径> --directoryperdb --serviceName MongoDB --install

mongod.exe --config “C:\MongoDB\Server\3.4\mongod.cfg” --directoryperdb --serviceName MongoDB --install

启动MongDB服务，在命令行中输入：

net start MongoDB

简单介绍一下使用的参数及其含义：

--install：安装MongoDB服务

--logappend:以每天为单位自动切割日志

--serviceName：安装Windows服务时使用的服务名

--serviceDisplayName：在Windows服务管理器中显示的服务名

--logpath：MongoDB日志输出文件名称。虽说该参数直译是“日志路径”，其实要指定的是一个具体的完整文件名。这里我使用的是C盘根目录下的MongoDB.Log文件。该文件不用事先创建，直接指定就是了。

--dbpath：指定MongoDB数据存放的路径。这个就是最关键的参数了，不仅该目录要存在，并且最好不要以“\”结尾。

--directoryperdb：这个参数很好理解，让MongoDB按照数据库的不同，针对每一个数据库都建立一个目录，所谓的“目录每数据库”

在bin中运行命令行

mongo

show dbs

use local

show collections

db.startup\_log.find()

**Mac安装MongoDB**

1) 安装神器 homebrew

brew.sh网站安装说明

安装成功后，在终端继续输入：

sudo brew update

2) 安装MongoDB

在终端输入一句，并等待完成

sudo brew install mongodb

启动命令

sudo brew services list

sudo brew services start mongodb

3) 运行MongoDB

在终端输入如下，每一句敲一次回车，注意空格

cd /

sudo mkdir -p /data/db

sudo chown -R david/data/db

再次要终端输入：

mongod

启动了MongoDB的服务，返回类似如下的信息，就表示成功了。

如果终端最下面出现一句，代表成功了。

waiting for connections on port 27017

如果有这样的提示：

WARNING: soft rlimits too low.Number of files is 256,should be at least 1000.

这样的提示无视就可以了，不需要解决，我们只是使用mongoDB用来开发，不需要升级。感兴趣的可以到stackoverflow查看解释。

<http://stackoverflow.com/questions/16621763/mongod-mac-os-x-rlimits-warning>

重新打开一个终端，输入

mongo

返回mongo的版本号和连接到的数据库，就安装好了。

**Linux下安装mongodb**

**1) 添加源，依次输入这三句**

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 -recv EA312927

echo “deb [http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2](http://repo.mongodb.org/apt/ubuntu%20trusty/mongodb-org/3.2) multiverse” | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list

sudo apt-get update

**2)安装最新版的mongodb**

sudo apt-get install -y mongodb-org

**3)启动mongod**

sudo service mongod start

**4)检查启动是否成功**

找到下面这个地址

/var/log/mongodb/mongod.log

如果看到waiting for connections on port 27017，就代表成功了

Linux sudo apt-get install mongodb

sudo yum install -y mongodb-org

## 课时25 Pycharm安装MongoPlugin插件

2. 安装第三方库pymongo

**Mac安装Pymongo**

#安装到python3

pip3 install pymongo

#安装到python2

pip2 install pymongo

**Linux安装pymongo**

sudo apt-get install build-essential python-dev

**Windows安装pymongo**

方法一：使用pycharm搜索pymongo安装

方法二：pip3 install pymongo

如果以上两种方法都安装失败，尝试手动安装

方法三：手动安装

1）首先需要安装C扩展

python3.5需要安装Visual Studio 2015

<https://www.visualstudio.com/downloads/download-visual-studio-vs>

注意：需要选择完整版的VS2010或2008，需要确保在Visual C++中勾选了“x64 Compilers and Tools”

2) 下载pymongo安装包

python3.5

<https://pypi.python.org/>

具体的安装wheel文件的方法在魔力手机第14页中查看。

**安装pycharm的MongoDB插件**

pycharm > preferences[设置] > plugins > browse repositories > 搜索mongo > 找到mongo plugin点击安装 > restart pycharm重启pycharm > 安装好之后，需要配置插件

view > 选中Tool Buttons 或点击最左下角的按钮> 就能在最右边边框里看到mongo explorer > 打开 > 选择齿轮和扳手，或者在设置中最下面的other settings中打开mongo servers > 点“+”添加服务器 > label中输入名称，url中输入localhost:27017

如果想让mongoDB工作，首先要启动mongDB服务器，mac中输入以下命令，并保证命令行不要关闭：

mongo

windows中已经添加mongodb为服务，会随windows启动而启动

**pycharm中mongoDB插件配置**

setting-->Plugins-->mongo

setting-->project: -->project interpreter-->pymongo-->install

## 课时26 MongoDB数据库导入导出

**数据导出功能**

确认mongo服务已经启动

show dbs

use local

show collections

db.startup\_log.find()

导出为json格式

mongoexport -d <数据库名称> -c <Collection名称> -o <json文件名称>

mongoexport.exe -d local -c startup\_log -o startup\_log.json

导出为csv格式

mongoexport -d <数据库名称> -c <Collection名称> --csv -f <字段名称> -o <csv文件名称>

mongoexport -d local -c startup\_log --csv -f hostname,startTime -o start\_log.csv

**mongodb数据导入功能：**

新建一个collection，把上面导出的数据导入到这个collection中。

mongo

db.createCollection(’log’)

show collections

new cmd

mongoimport.exe -d local -c log --file startup\_log.json

--headerling忽略第一行，也就是不导入表头

mongoimport.exe -d local -c log --type csv --headerline --file startup\_log.csv

## 课时27 第一节课程 开始使用数据库MongoDB

课程回顾

这节课我们把瓦尔登湖的文本导入到了数据库中，并做了一些统计

在这个过程中，理解了MongoDB就像是我们熟悉的Excel表格

给数据库命名，就相当于是给Excel表格起名字

walden = clent[‘walden’]

创建表单，就像是在Excel中创建sheet

sheet = walden[‘sheet’]

往数据库中写入数据，就像是在Excel中填写每行数据

sheet.insert\_one(data)

数据库的操作符，其实很好记

$lt less than <

$lte less than equal <=

$gt greater than >

$gte great than equal >=

$ne not equal !=

## 课时28 第一节练习项目 在MongoDB中筛选房源

<http://bj.xiaozhu.com/search-duanzufang-p1-0/>

爬取小猪短租这个页面的前三页，把爬取到结果存储到MongoDB数据库中。

然后在数据库中筛选出所有价格大于等于500元的房源，并打印出来。

MongoDB数据库基本操作

1。使用mldn数据库

use mldn

2。创建一个表

db.createCollection(“emp”)

这个时候mldn数据库才会真正存在

db.emp.find()

db.dept.insert({“deptno”: 10, “dname”: “账务部”, “loc” : “北京”});

show collections

发现dept的表自动创建

3。查看emp表的数据

语法：db.集合名称.find({若干条件})

db.dept.find()

4。增加不规则的数据

var deptData = {

“deptno”: 20,

“dname”: “研发部”,

“loc”: “深圳”

“count”: 50

“avg”: 8000.0

}

db.dept.insert(dptData)

db.dept.find()

## 课时29 第一节练习项目详解

从商品列表页面得到所有的商品名称和价格，插入数据库，然后查询数据库，过滤得到符合条件的数据

第一步：获取的信息写入数据库

第二步：从数据库获取符合条件的信息

pycharm新建python文件

新建以main命名的python文件

梳理思路，写好程序流程和核心函数

打开main.py，写好基本信息，定义两个函数，一个运行入口，一个获取详情页面的信息

#!/usr/bin/env python

#-\*- coding: utf-8 -\*-

import pymongo, requests

from bs4 import BeautifulSoup

#把信息插入数据库

def insert\_fangzi\_info(url)

pass

#从数据库获取信息

def find\_fangzi():

pass

写好连接数据库的代码

通过pymongo库连接MongoDB，数据库名字为xiaozhu，表名为fangzi

*#连接MongoDB，数据库地址为：localhost，端口号为27017*client = pymongo.MongoClient(**'localhost'**,27017)  
*#从MongoDB中选择名为xiaozhu的数据库*xiaozhu = client[**'xiaozhu'**]  
*#从xiaozhu数据库中选择名称为fangzi的表*bnb\_info = xiaozhu[**'bnb\_info'**]  
  
**def** insert\_faingzi\_info(url):  
 **pass  
  
def** find\_fangzi():  
 **pass**

写爬取列表页面代码

写解析列表页面的代码，先requests库获取网页数据，再BeautifulsSoup库解析网页数据

def insert\_fangzi\_info():

#请求列表页面，获取页面数据

wb\_data = requests.get(url)

#开始解析数据，采用lxml解析引擎

soup = BeautifulSoup(wb\_data.text,’lxml’)

根据css selector，提取相应信息列表

#soup.select得到的是列表，变量名最好是复数形式

titles = soup.select(‘span.result\_title’)

prices = soup.select(‘span.result\_price > i’)

循环标签列表，得到标签信息

for取出列表中的标签，通过get\_text()函数得到标签内容，然后把信息插入数据库

#通过for取出列表中的标签，然后通过get\_text()获取标签内的文本

for title, price in zip(titles, prices):

#price.get\_text()得到的是字符串，需要int()函数转成数字类型

info = {

‘title’: title.get\_text(),

‘price’: int(price.get\_text())

}

fangzi.insert\_one(info)

写从数据库查找数据，过滤符合条件的数据

for取出数据库中得到的数据，这里用fangzi.find()函数获取，表示从名称为房子的表中查找数据

def find\_fangzi():

#从xiaozhu数据库的fangzi表中用find()函数查询所有数据

for info in fangzi.find():

#info我们插入的数据都有title和price，取出每条信息的price，用来比较

if info[‘price’] >= 500:

print(info)

分析列表页链接规律

打开<http://bj.xiaozhu.com，点击第2>页，第3页，观察地址的规律

[http://bj.xiaozhu.com/search-duanzufang-p{}-0/](http://bj.xiaozhu.com/search-duanzufang-p%7b%7d-0/)

替换括号所在位置为页码对应数字，数字为几则是第几页

生成三个列表页面链接

#根据规律，生成3个列表链接, 结束数字要比最后列表中的数字大一

urls = [‘http://bj.xiaozhu.com/search-duanzufang-p{}-0’.format(number) for number in range(1, 4)]

#从链接列表中，用for一个个取出来

for single\_url in urls:

#把得到的列表页面链接，传给函数，这个函数提取房子的标题和价格

insert\_fangzi\_info(single\_url)

#数据插入数据库后，查找符合条件的数据

#定义这个函数不要缩进，否则就变成了每插入一条取一次了

find\_fangzi()

## 课时30 第二节课程 爬取大规模数据的工作流程分析

为了爬取58的30万数据做工作流的准备

第一步：观察页面特征

在以前的课程中是针对特定的页面进行了信息元素的抓取，但对于大规模数据的抓取，还要去分析我们的程序是否具有通用性，能否抓取到所有页面的信息。观察页面特征得到边界条件和局限性。

不同页面不规则的问题

如有的页面中就有借钱买的功能，就把原来的价格信息覆盖掉了

58同城的分页问题

如二手手机号页面，与其它页面的结构完全不同。

因为后期要做数据分析，尽量把数据的结构保持完整和一致，一些不靠谱的数字也要清洗掉。

最后一页的问题

对于不同的页面，爬取到最后一页的时候，会报错，这种错误也是需要进行避免。

第二步：设计工作流程

设计合适的工作流程，保证抓取的过程是高效的。在程序运行的过程中，是由不同的函数，各司其职，相互配合完成整个工作。

spider1: 从列表页获取商品的url，保存到mongoDB的数据库tongcheng中的一个url\_list的工作表中。

spider2: 从url\_list工作表中依次取出保存的商品的url，依次访问详情页，获取商品的信息，存储到tongcheng数据库中的item\_info工作表中。包括发布时间

为什么不设置爬虫在列表页中获取一个商品的url就去访问详情页，进而得到商品的详细信息呢？做这种大规模的数据爬取，最好一次只做一件事情，这样会大大提高我们程序的稳定性。例如如果程序在爬取商品详情时出现了问题，那至少我们已经获取到了所有商品的url信息，这样再次进行任务时就不用从零开始了。

首先要取到各个频道的链接

http://bj.58.com/sale.shtml

这个网页中导航信息中的所有链接爬取下来。

channel\_extract.py

4。创建用来计数的监控程序

counts.py

import time

第二节练习项目 爬取手机号

抓取58同城中的手机号类目下的所有帖子的标题和链接，存储在数据库中

<http://bj.58.com/shoujihao/>

仿照视频中的安全，把工作流中的爬虫2设计出来并成功运行，然后存储在数据库中。

实现思路，从数据库中取出url后一次访问页面得到数据并进行存储

## 第二节练习项目详解 爬取手机号

第一步：观察页面特征

1。观察手机号列表页面的链接规律

2。观察列表页面内容，判断列表页面的最后一项

3。观察详情页面内容，去掉不合要求的详情页面

4。观察详情页面内容，提取详情页面重要信息的css path

第二步：设计工作流程

代码爬取流程：先到列表页面提取详情页面链接，然后进行详情页面提取详细信息

观察列表页面链接特征

打开<http://bj.58.com/shoujihao/，点击第2>页，第3页，观察地址信息，分析规律

根据规律，多次替换链接中的页码，查看返回的页面内容差异

<http://bj.58.com/shoujihao/pn1000>

中的内容很少，最下面有共0条信息，可以据此判断最后的页码，为0，停止分析。

提取列表页面信息的css path，分别提取标题，链接和页面询问的手机号数目。

观察详情页面内容特征

浏览器从列表页面打开多个详情页链接，发现有些详情页不正常，不能提取对应信息，根据url的不同，对其进行筛选。

pycharm新建py文件

写好代码的大致结构

先定义两个函数，一个列表页面提取链接，一个从详情页面获取信息

多进程爬虫数据抓取

thread process线程与进程

课程回顾

这节课我们终于使用多进程爬取了58二手商品的十万级数据

我们理解了，单进程单线程就像是一个人在一张桌子上吃饭

单进程多线程，是多个人在一张桌子上一起吃饭

多进程单线程，就是多个人，各自在一张桌子 上吃饭

多进程多线程是效率最高的，多个人在多张桌子 上一起吃饭

我们学习使用了multiprocessing这个库轻松搞定了多进程。

pool=Pool()

还添加了一个小程序，帮我们数数

while True:

print(url\_list.find().count())

现在，你已经可以爬取大规模数据了。

来挑战本周的大作业，爬取赶集的所有二手商品数据

## 第三周练习项目 设计断点续传程序

假设在抓取过程中会遇到网络问题而导致程序停止，在程序中设计一个功能，保证数据库中开始抓取的数据不会重复。

对爬取的信息进行处理，所有url相同的认为是同一个商品，只保留一个。

上一周的大作业，我们爬取了一页的二手商品信息，那么这一周，我们用所学知识来爬取二手市场的全部信息，数据量在10W这个数量级。

课程中已经爬取了58同城，那么我们在作业里面爬取它的兄弟网站：赶集网

需要爬取赶集网-北京-二手市场的所有类目的商品信息

<http://bj.ganji.com/wu>

点进来，拉到页面底部，所有加粗的类目都是，能看到一共20个。

点进来的列表页，需要爬取个人类目下的全部帖子。

对于每个帖子，需要爬取这些信息

商品标题

发帖时间

类型

价格

交易地点

新旧程度

也就是说，赶集网-北京-二手市场的所有帖子都需要爬取，除了某些不规则 的页面可以不包括在内。建议使用多进程的方式爬取。

## 第二周补充资料

MongoDB教程(中文)

<http://www.runoob.com/mongodb/mongodb-tutorial.html>

解读：课程里面主要是理解MongoDB的最基本操作，关于增删改查以及一些高级用法，都可以在教程里面查到。

MongoDB的操作集合(英文)

<https://blog.codecentric.de/files/2012/12/MongoDB-CheatSheet-v1_0.pdf>

解读：这里面是一些主要操作的语法汇总，想不起来的时候，到这里来查阅就好。

## 第三周：数据统计和分析

收集了数据之后，我们通过统计和分析，去发现数据的价值

课前准备

这周课程需要安装2项：

1。安装网页端编译器 Jupyter Notebook，也叫Ipython notebook

2。安装数据可视化的库highchart

1。安装Jupyter Notebook

无论是windows,mac还是linux，都可以使用pip来安装。

打开终端或者命令行，如果想安装到python3中，需要输入

pip3 install jupyter

如果你安装了python2和3两种版本，可能会遇到安装目录的问题，可以换成

python3 -m pip install jupyter

如果想要安装到python2中，需要输入：

pip install jpyter

Windows用户需要注意

可能会出现这样的报错

building ‘zmp.libsodium’ extension error:[WinError 2]系统找不到指定的文件

解决方案是安装库libsodium，还是使用pip安装即可

pip3 install libsodium

另外，如果选择手动下载压缩包安装，地址是<https://download.libsodium.org/libsodium/releases>

linux用户需注意

Ubuntu用户可能会遇到需要安装Python C扩展的问题，这时候需要输入：

apt-get install build-essential python3-dev

启动jupyter Notebook

安装好了之后，在命令行或者终端输入

jupyter notebook

然后会跳转到jupyter的网页，上面显示了你本地的python文件，这样就启动成功了。

2。安装库highchart

pip安装highchart

同样，无论windows，mac还是linux，都可以使用pip来安装。

打开终端或命令行，如果想安装到python3中，输入：

pip3 install charts

如果安装了python3和2两种版本，可能会遇到安装目录的问题，可以换成

python3 -m pip install charts

如果想安装到python2中，需要输入：

pip install charts

安装成功后，需要验证在jupyter notebook里面能否正常使用。因为python3很可能会报错。

首先，启动jupyter Notebook。在命令行或终端中输入

jupyter notebook

会跳转到jupyter的网页，点击new > python 3

在新页面里面输入

import charts

然后点击cell > Run

可能会出现ImportError: No module named ‘plot’这样的报错。

这时候需要做这样2步：

1）找到charts所在的文件目录。

再到jupyter notebook打开一个新的python文件，还是按照刚才的方法输入并运动

import sys

print(sys.path)

会看到类似这样的安装路径

在其中找到site-packages的路径：‘/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages’ 每个人的路径不同，以自己找到的为准。

然后不同系统的操作不太相同

windows用户直接按照路径目录打开site-packages，找到charts的文件夹并打开。如果找不到，就在python3.5这个文件夹目录下搜索一下。

linux用户使用终端，按照目录，找到charts的文件夹并打开。

mac 用户使用finder中的前往来进入目录。

在里面找到charts的文件夹，并打开

2）替换charts文件夹中的7个文件。

charts替换文件地址。

<https://github.com/mugglecoding>

用这7个文件去替换刚才site-packages里面的charts文件夹中同名的7个文件。

然后再到jupyter的文件里面输入并运行：

import charts

看到类似Server running in the folder /users/linqiangqian at 127.0.0.1:55280这样的提示，就表明成功了。

注：手册可能无法穷举所有问题，其他的安装问题或使用说明可以查看官网：

jupyter notebook官网：<http://jupyter.org>

highcharts官网：<http://www.highcharts.com>

## 如何让数据说话

麦当劳 80%的利润来自汉堡

如何让数据说话，可以分为三个步骤：

第1步：提出正确的问题

有多少家庭主妇购买黄豆罐头？

科学的广告 我的广告生涯

汽车潜在用户调制鸡尾酒的能力？

商业冒险：华尔街的12个经典故事。

正确的提问能解释现象

错误的提示却强行关联无关的事物。

AB页面哪种效果更好？

facebook广告的方案，共45种方案选择。

为什么新版本的效果更好？

这种方法经常在公司中用来进行数据造假。

正确的提问是验证假设

错误的提问是证明自己是对的。

豌豆夹的机型分布数据

p2p的羊毛党数据。

正确的提问是通过数据探索方向

错误的提问是没有提出问题，有了数据却不知道应该做什么。

总结：如何提出正确的问题。

解释现象而不是强行关联。

验证假设而不是证明自己是对的。

探索方向而不是为了数据而数据

第2步：通过数据认证，寻找答案。

提出了正确的问题之后，第二步就是通过数据谁，寻找问题的答案。

方法：

对比 详情页转化率。一种是横向对比。与其它产品对比 。第二种是纵向对比，自己的产品不同时间的对比。

细分 提出的问题可能是一个大的目标。要把这个目标拆分为一个个小的问题。如电商领域：销售额 = 日活跃\*购买转化率\*客单价

溯源 找到了问题之后，再去定位到源日志。如销量异常的问题。如游戏玩家流失的问题。15级玩家流失的比较多，15级boss任务太难。

第3步：解读数据，回答问题。

这一步也是数据会说话还是数据会说谎的区别

知乎 数据会说谎的例子。

数据不会出错，人会出错。数据不会说谎，人会说谎。

如何去避免错误。

样本问题。

样本是存在偏差的。如在微博上一个‘你是否经常使用微博’的投票。

因果关联错误。

支付宝改版会缴费激增的问题。

facebook失去了希腊债务危机？

忽略前提

有些数据结论是在一定的前提下符合某种特定场景下得出的。但解读者有意无意忽略前提，将结论扩大化，显著误读。这种错误经常出现在朋友圈转发的各种谣言。

2015年互联网融资泡沫。

第1步：提出正确的问题

2015年互联网的融资情况怎么样？

第2步：数据认证。

数据源：itjuzi.com两万多家互联网公司。

对比。对比2014年。

细分。对问题的细分。

溯源。

第3步：解读数据

样本偏差。2万多家公司，但并不会包含所有的公司，而是只有知名的公司。所以这里反映的情况应该比实际的互联网融资情况好一些。

七日内北京各城区发帖量。

## 第一节练习项目

上一周的大作业中，我们爬取了赶集的二手商品数据。为了更好的了解二手市场的行情，可以利用这些数据做哪些分析呢？请按照课程中的步骤设计分析目标。

1、提出的问题是什么

2、通过数据认证，可以通过细分、对比、溯源的方式来统计哪些数据项

3、解读数据的时候，有哪些需要注意的地方

## 第二节 开始做简单的数据图表

七日内二手物品市场不同地区的发帖数量

第1步：整理清洗数据

第2步：更新数据库

第3步：数据的可视化

jupyter notebook可以理解成一个加强版的交互式的网页端的编译器，比终端更智能更方便。

点tab键，会自动提醒，点车，换行。shift+回车是执行一个cell中的代码。每一行的数据都能滑动，双击会折叠。

# 导入数据须知

1. 首先运行 mongo shell在数据库中创建一个 collection, db.createCollection('the\_name')

2. 接下来打开一个新的终端直接在终端/命令行中使用命令导入json 格式的数据 mongoimport -d database\_name -c collection\_name --type json --file path/file\_name.json

mongoimport -d ceshi -c tongcheng --type json --file D://David/Desktop/fourweeks/Plan-for-combating-master/week3/week3\_homework/data\_sample/sample.json

调用模块

import pymongo

连接数据库

client = pymongo.MongoClient(‘localhost’, 27017)

ceshi = cleient[‘ceshi’]

item\_info = ceshi[‘item\_infoX’]

for i in item\_info.find().limit(300):

print(i[‘area’])

只想看地区的字段。地区是一个二级结构，键值后面对应一个列表，列表中包含多个值。包括-，包括None。把None替换为[不明]，把标点符号去掉

引用标点符号的库。

from string import punctuation

对数据进行整理

for i in item\_info.find().limit(300):

#if i[‘area’]相当于if True，也就是说如果它存在。

if i[‘area’]:

#替换标点符号的方法称为‘列表的解析式’，原来的area是一个列表，替换之后还是一个列表。

area = [i for i in i[‘area’] if i not in punctuation]

#如果它不存在，也就是None，还把它装在一个列表中。

else:

area = [‘不明’]

print(area)

2、在数据库中更新整理后的数据

第2步：更新数据库

之前学过mongoDB中的insert方法

这里用到update方法

db.collection.update()

db为数据库名称，在编辑器中可以不写，但在命令行中必须要写上。

update可以接收2个参数。

update(要更新的文件, 怎么更新或更新哪里)

{id:1, name:0, info:3}

找数据库中所有id==1的记录。

update({id:1}, {$set:{name:2}}) #$set为操作符，set后面{}中的内容代表要进行的操作。如果name存在，就把它改为2，如果不存在，就添加一个新的字段并赋值。

修改数据库之前要先备份数据库

新建terminal

mongod

mongo

show dbs

use ceshi

show collections

show tables

在测试之前，先要建立一个空的表，把已有的数据移过去。

db.createCollection(‘item\_infoY’)

#把已有的item\_infoX数据库复制到新建的item\_infoY空的表中。

db.item\_infoX.copyT(‘item\_infoY’)

如果出现‘warning: db.eva1 is deprecated‘表明复制成功。

修改连接的数据库名称

item\_info = ceshi{‘item\_infoY’}

再重新运行上面的每一行代码修改的才能有效。

for i in item\_info.find():

if i[‘area’]:

area = [i for i in i[‘area’] if i not in punctuation]

else:

area = [‘不明’]

item.info.update({‘\_id’: i[‘\_id’]}, {‘$set’: {‘area’: area}})

打印一下结果

for i in item\_info.find().limit(300):

print(i[‘area’])

3、数据可视化图表

import charts

每一次修改都要再次运行。

如果出现server running in the folder …就说明运行成功了。

一个小例子，data=[11]，必须要把数字用[]括起来，因为highcharts是用js写的一个程序。要按它的规则去写。type:柱状图。必须要有name,data,type至少这三种，可以查看更多的实例。

[www.highcharts.com/demo/column-stacked](http://www.highcharts.com/demo/column-stacked)

series = [{

‘name’: ‘OS X’,

‘data’: [11],

‘type’: ‘column’

},{

‘name’: ‘Ubuntu’,

‘data’: [8],

‘type’: ‘column’

},{

‘name’: ‘Windows’,

‘data’: [12],

‘type’: ‘column’

},{

‘name’: ‘Others’,

‘data’: [29],

‘type’: ‘column’

}]

series2 = [{‘name’: ‘John’, ‘data’: [5], ‘type’: ‘column’},{‘name’: ‘John’, ‘data’: [5], ‘type’: ‘column’}]

show=’inline’, 指定显示的图表是在当前页，而不是新建一个窗口。

charts.plot(series, show=’inline’, option=dic(text=’Charts are AWESOME!!!’))

4、为数据统计项，调整数据结构

想要得到7日内的发帖量，也就是7日内有多少人想要交换2手物品。已经得到了地区，地区有重复。要统计每一个区域重复的数量，就能得到每一个区域的发帖量。

把所有地区的列表进行去重，找出一个索引来，才能统计出各个区的发帖次数。

#新建一个列表保存所有重复的区域

area\_list = []

for i in item\_info.find():

#取[0]，即北京的每个区

area\_list.append(i[‘area’][0])

#area\_index用来计数，先用set方法把它变成一个集合。集合是没有重复元素的，所有重复的区都会合并为一个，再把集合转化为列表。就得到了索引。

area\_index = list(set(area\_list))

print(area\_index)

post\_times = []

for index in area\_index:

post\_time.append(area\_list.count(index))

print(post\_times)

需要自动生成highchart的数据结构

生成器，由于js的代码只能识别name,data,type这三个字段，所以我们也必须使用这三个字段。

def data\_gen(types):

length = 0

if length <=len(area\_index()):

for area,times in zip(area\_index,post\_times):

data = {

‘name’: area,

‘data’: [times], #times必须要放在[]中。

‘type’: types,

}

yield data #让它不断的生成

length +=1

因为它是一个可生成器，所以不能像函数那样去直接调用。

for i in data\_gen(‘column’):

print(i)

使用真实的上海或北京地区列表，而忽略其它的。

用一个列表解析式

series = [data for data in data\_gen(‘column’)]

charts.plot(series, show = ‘inline’, options = dict(title = dict(text=’七日内北京城区二手物品发帖量’)))

为了让结果更加可靠，需要选择多个7日内的发帖量进行平均的期望值的计算。和数据的收集量有关系，如果想要得到一个准确、科学、客观的数据，需要进行长期的数据的收集。还要构建一个合理的数学模型进行计算。

本节回顾：

这节课，我们用最简单的方式完成了一次完整的数据分析过程。

我们开始使用python神器jupyter notebook

通过整理数据，把数据格式变成了我们想要的样子。

学习了在MongoDB数据库中更新数据的方法

为了实现统计项，调整数据结构

还学习了使用迭代器，让它替我们自动调整数据结构。

这样的方式实现，容易理解，不过写起来稍微有点麻烦，有没有更简洁的实现方案呢？

N日内指定地区的发帖量的变化曲线。

## 第二节练习项目：绘制各类目对比柱状图。

统计赶集网-北京-二手市场19个大类目的发帖量，手机号码不要。

并在jupyter notebook中绘制出各类目发帖量对比的柱状图

源数据来自第二周实战作业时跑的结果。可以使用源代码中的数据样本week3\_homework > data\_sample

## 第三节课程 使用find()函数精确查找数据。

N日内指定地区的发帖量的变化曲线。

第1步 整理清洗数据

第2步 更新数据库

第3步 数据的可视化

db.collection.find()

find不会修改原来的数据结构，只会在原数据中查找并显示出来给我们。

可以传入2个参数，第一个参数决定返回的数据，第二个参数决定返回的数据的展现方式，可以对数据进行过滤。

find(找什么文件，返回的文件的格式)

{id:1001, name:david, info:32, cate:12000}

name:1, info:1并不是查找name ==1, info ==1的值，而是True的意思，只显示这两项。

第1个参数指定查找的内容的key和value值，第2个参数指定显示的内容。

find({id:1001}, {name:1, info:1})

{id:1001, name:david, info:32}

import pymongo

client = pymongo.MongoClient(‘localhost’, 27017)

ceshi = client[‘ceshi’]

item\_info = ceshi[‘item\_infoY’]

for i in item\_info.find({},{‘area’:1}).limit(300):

print(i)

#不显示数据库中的id值，只显示area的字段。

for i in item info.find({},{‘area’:1,’\_id’:0}).limit(300):

print(i)

update函数中有一个$set的操作符，操作符指明了不仅选择哪个，还对哪个进行一个特定的操作。find函数中也有类似的操作符。

地区中还是有主地区，次级地区，可以用find的切片方法只查找主地区。$slice:1分片操作返回所有可被分片的数据中的第一个元素。与python中操作列表的方法相似。但此时其它未说明显示还是不显示的字段就全部显示出来了。这是mongoDB中的特殊设定。一旦设定了某一字段中具体的显示方式，就必须要逐一的指定其它所有的字段的显示方式。如：

for i in item info.find({},{‘area’:{‘$slice’:1},’\_id’:0}).limit(300):

print(i)

for i in item info.find({‘pub\_date‘:’2016.01.12’}, {‘area’: {‘$slice’:1}, ’\_id’: 0,’ price’: 0, ’title’: 0}).limit(300):

print(i)

想要查找在2016年01月12和2016年01月14日的发帖量，$in称为条件操作符

for i in item\_info.find({‘pub\_date’: {‘$in’: [‘2016.01.12’, ‘2016.01.14’]}}, {‘area’: {‘$slice’: 1}, ‘\_id’:0, ‘price’: 0, ‘title’: 0}).limit(300):

print(i)

options = {

‘chart’: {‘zoomType’:’xy’},

‘title’: {‘text’: ‘Monthly Average Temperature’},

‘subtitle’: {‘categories’: [‘周一’,’周二’,’周三’,’周四’]},

‘yAxis‘: {‘title’: {‘text’: ‘数量’}}

}

series = [{

‘name’: ‘OS X’,

‘data’: [11, 2, 3, 4],

‘type’: ‘line’

},{

‘name’: ‘Ubuntu’,

‘data’: [8, 5, 6, 7],

‘type’: ‘line’

},{

‘name’: ‘Windows’,

‘data’: [12, 6, 7, 2],

‘type’: ‘line’

},{

‘name’: ‘Others’,

‘data’: [29, 24, 58, 23],

‘type’: ‘line’

}]

charts.plot(series, show=’inline’, options=options)

获得某一格式数据的思路，不同的数据结构，对其进行整理的思路都不尽相同，但思路都是相似的。

已知：

日期(2015.12.27-2016.12.27)

区域名称(朝阳, 海淀)

求解：

单日内地区发帖量

1。让程序明白时间段，获得单日日期

2。使用find函数查找指定数据

3。转化数据

1、使用find函数筛选并整理数据字段

for i in item\_info.find().limit(300):

print(i)

for i in item\_info.find({}, {‘\_id’: 0, ‘pub\_date’: 1}).limit(300):

print(i)

发现日期有两种格式，一种是2016-12-17，一种是2016.12.17，统一一下格式。用-对日期进行分割，如果没有-，也会进行分割，但只有一个元素。如果有None的话，还要进行一个判断。

for i in item\_info.find():

frags = i[‘pub\_date’].split(‘-’)

if len(frags) == 1:

date = frags[0]

else:

date = ‘{}.{}.{}‘.format(frags[0], frags[1], frags[2])

#print(date)

item\_info.update\_one({‘\_id’: i[‘\_id’]},{‘$set’: {‘pub\_date’: date}})

#这样写是有些冒险的，因为没有考虑到None的结构，为了保险起见，还要再进行一次判断。

for i in item\_info.find({}, {‘\_id’: 0, ‘pub\_date’: 1}).limit(300):

print(i)

2、让程序知道发帖时间，尤其是时间范围

相当于把日期转化为程序可以理解的语言。

from datetime import timedelta, date

a = date(2015,5,10) #把输入的整数转变为日期字符串

print(a)

2015-05-10

d = timedelta(days = 1)

print(d)

1day, 0:00:00

设计一个转换时间的函数。把两个日期相加或相减，加上中间差的日期，打印出中间间隔的时间段。

def get\_all\_dates(date1, date2):

the\_date = date(int(date1.split(‘.’)[0], int(date1.split(‘.’)[1], int(date1.split(‘.’)[2]))

end\_date = date(int(date2.split(‘.’)[0], int(date2.split(‘.’)[1], int(date2.split(‘.’)[2]))

days = timedelta(days=1)

#要让日期能够持续的返回，返回所有起始日期和结束日期之间的所有日期

while the\_date <= end\_date:

#只要以上的条件成立，就按按照Y.m.d的方式不断的返回日期，每一次返回起始日期都会增加一天

yield the\_date.strftime(‘%Y.%m.%d‘)

the\_date = the\_date + days

#用一个for循环调用函数

for i in get\_all\_dates(‘2015.12.24’, ‘2016.01.05’)

print(i)

作为一个程序员，我的工作是向计算机解释人们想要它做的事情。

3、根据已知，求解未知信息

设计主函数，实现单日内某地区发帖量的函数

#要想得到单日内某地区发帖量，用find函数来实现，item\_info.find({‘pub\_date’: date, ‘area’: area})，date是上面get\_all\_dates中得到的日期区间中的任意一天，这就需要一个日期的循环，调用get\_all\_dates这个函数就可以实现，而此函数需要传入起始日期和结束日期两个参数。area是某个地区，想要得到所有地区的数据，需要一个地区的循环。故此主函数需要传入三个参数，两个日期和一个地区。

item\_info.find({‘pub\_date’: date, ‘area’: area})对于某一特定的日期和地区来说，是找到这一天中这一地区所发的所有帖子，可以把它列表化，只要求出这个列表的长度，就得到了这个地区在某一天中的发帖量。

def get\_data\_within(date1,date2,areas):

#地区的循环

for area in areas:

#日期的循环

for date in get\_all\_dates(date1, date2):

a = list(item\_info.find({‘pub\_date’: date, ‘area’: area}))

print(‘#’\*20, date, area, len(a), ‘#’\*20)

#调用函数

get\_data\_within(‘2015.12.24’, ‘2016.01.05’, [‘朝阳’])

4、调整数据格式

还要按照highchart的固定格式进行格式化

def get\_data\_within(date1,date2,areas):

for area in areas:

area\_day\_posts = []

for date in get\_all\_dates(date1, date2):

a = list(item\_info.find({‘pub\_date’: date, ‘area’: area}))

each\_day\_post = len(a)

area\_day\_posts.append(each\_day\_post)

data = {

‘name’: area,

‘data’: area\_day\_posts,

‘type’: ‘line’

}

yeild data

#因为函数中有生成器，不能直接调用函数。

for i in get\_data\_within(‘2015.12.24’, ‘2016.01.05’, [‘朝阳’]):

print(i)

for i in get\_data\_within(‘2015.12.24’, ‘2016.01.05’, [‘朝阳’, ‘海淀’, ‘通州‘]):

print(i)

5、进行数据的可视化

#options是一个二级字典结构

options = {

‘chart’: {‘zoomType’: ‘xy’},

‘title’: {‘text’: ’发帖量统计’},

‘subtitle’: {‘text‘: ‘可视化统计图表’},

‘xAxis’: {‘categories’: [i for i in get\_all\_dates(‘2015.12.24’, ‘2016.01.05’)]},

‘yAxis‘: {‘title’: {‘text’: ‘数量’}}

}

series = [i for i in get\_data\_within(‘2015.12.24’, ‘2016.01.05’, [‘朝阳’, ‘海淀’, ‘通州‘])]

charts.plot(series, show=’inline’, options=options)

得到了发帖量的走势。但这个走势只有在数据量足够大的时候才有意思，这里也只能作为一个演示而已。

还可以放大，缩小。

课程回顾：

这节课，我们用find函数做了复杂一点的数据分析

find函数有它自己的规则，掌握之后你就会爱上它

折线图的数据结构，和之前的柱状图很相似，只是type稍有不同

还有一种实现方式，只需要4行代码，运行效率是现在的5倍

某一天二手物品在随后的七天内，交易时长为1天的类目分布占比。

## 第三节练习项目 绘制发测试贴量折线图

统计二手手机，二手笔记本，台式电脑/网络 这3个类目在最近7日内，发帖量随时间的变化情况，并用折线图绘制出来。

## 第四节课程，使用聚合管道高效查找数据

某一天二手物品在随后的七天内，交易时长为1天的类目分布占比。用饼图的方式，显示用1天时间，卖出去最多的是哪个类目。也就是统计出某个帖子在发出来后到帖子被删除的时间。帖子被删除就可以认为是物品被交易出去了。从对交易出去的天数的统计就可以对公众的需求有一个大致的了解。也可以对交易时长进行统计和分析，分别分析每一种物品在第一天，第二天，第三天等交易出去的数量的分布。这就要求对网站的数据进行长期的抓取和每日的监测。至少7天时间。每天固定的时间对所有的数据进行监测，看它是否被交易出去了。

collection.aggregate(pipeline)

database -->pipeline--> data

管道模型，就是从数据库中，通过管道，得到我们想要的数据。

就像一个管道，通过一层层截流的方式得到数据

不同的管道有不同的作用，有的管道负责筛选和匹配数据，找出我们需要的数据，有些管道就负责聚合整合所有的统计，还有些管道可以进行重新分组和重新命名的工作。不同的管道连接起来，最后就能够得到我们想要的数据。管道pipline是一个列表，列表中有许多过滤管道，每一个管道都是类似字典的结构。共有13种不同功能的过滤管道，常用的是前4种。

pipeline = [

**{$match: ?},**

**{$group: ?},**

**{$sort: ?},**

**{$limit: ?},**

{$skip: ?},

{$unwind: ?},

{$redact: ?},

{$sample: ?},

{$out: ?},

]

先看一下作出想要的饼状图应有的数据结构

for i in item\_info.find().limit(300):

print(i)

其中多出来一个time的字段。在一周的时间里，每天去监测这个页面，如果这个页面消失了，是404的页面，就记录它消失的时间，用交易的时间减去发布的时间，就得到一个交易时长。time: 0表示在7天内都没有被交易出去。一开始都默认把所有的time设为0，如果它交易了，再用update函数把它为交易的时间。

抓取联系方式，QQ号，手机号，再分析同一个联系人所发的商品，看是不是一个商品多次发帖，如果是的，只保留最新的帖子。也可以用这种方式来排除虚假的交易。到后期对图片进行识别，不同帖子中如果存在完全相同的图片也视为同一个帖子。

再排除掉商家的帖子，只抓取个人的帖子。

pipline是一个列表，在这个列表中实际上是用管道的方法实现了多层过滤。先写它的match过滤，它和find()函数中第一个参数的用法很像。第二层字典功能就定义了想要匹配的元素的键值特征。在这里想要匹配发布日期，想要所有在2015年12月24日发布的商品。

pipeline = {‘$match’: {‘pub\_date’: ‘2015.12.24’}}

然后把pipline作为参数传入到agrregate函数中，还是需要用循环的方式才能返回结果。

for i in item\_info.agrregate(pipeline):

print(i)

match还可以用来同时筛选多个条件。例如找出在2015.12.24日发布并在3天交易出去的数据。

pipeline = {'$match':{'$and':[{'pub\_date':'2015.12.24'},{'time':3}]}}

for i in item\_info.agrregate(pipeline):

print(i)

还可以使用比较运算greater than, less than等，找出在一定日期范围内发布的商品。

group管道

是用来把数据中出现的次数进行加和，乘除和求平均等计算型的分组。可以认为group像一个函数一样，接收两个参数，第一个参数‘\_id‘告诉程序以什么来进行分组，要找的内容是数据库中已有的一个字段，这一点与其它的操作都不一样。，第二个参数是我们自定义的一个字段，要对进行分组的字段进行一种统计型的操作，一般是加和的操作。如sum:1表示每找到1个就加1。当然也可以加2，加3等等。这里的意思是对price进行分组，相同price的就加1。这里对price进行分组本身并没有太大的意义，只是来演示一下group的用法。

前两节课中统计某一个商品出现的次数，用的是python的循环的方法把它找出来了，但相对于pipeline这种方法，效率会低很多。

pipeline = [{'$match':{'$and':[{'pub\_date':'2015.12.24'},{'time':3}]}},

{'$group':{'\_id':'$price','counts':{'$sum':1}}}

]

for i in item\_info.agrregate(pipeline):

print(i)

sort管道。按照某一字段以某一顺序进行排序。以上面的counts为依据进行分组。counts: 1表示按照升序排列，-1表示倒序排列。

pipeline = [{'$match':{'$and':[{'pub\_date':'2015.12.24'},{'time':3}]}},

{'$group':{'\_id':'$price','counts':{'$sum':1}}},

{‘$sort‘: {‘counts’: 1}}

]

limit管道。和find函数后面的limit用法是一样的。截取前10个。

pipeline = [

{'$match':{'$and':[{'pub\_date':'2015.12.24'},{'time':3}]}},

{'$group':{'\_id':'$price','counts':{'$sum':1}}},

{'$sort' :{'counts':-1}},

{'$limit':10}

]

# {'pub\_date':'2015.12.24'}

下面就用4个管道一次性的处理过滤我们的数据。

看一下想要得到的数据。例子中得到的是发帖数量和分类之间的关系。而我们想要得到每天的交易量与种类之间的关系。

首先要找到的是，某一天发布的物品在我们监测的7天内交易时长为一天的数量，也就是发布的物品在发布后一天内被交易出去的数量。所以有两个参数，一个是时间的区间长度，即是7天。另一个是物品发布后到交易的时间长度，是1天。

而类目呢，还有一个问题。类似于area，它也是一个多级的结构，在一个列表中存储着四级的结构，只有第3级才是我们想要的结构，即在最初抓取的15个大类，故要进行处理一下。使用slice分片操作来进行过滤。操作一个已有的类目时，也需要在前面加上一个$符号。'$slice':['$cates',2,1]表示对cate字段进行分片操作，并且跳过前两个元素，取跳过前两个元素后的1个元素，也即取第3个元素。

for i in item\_info.find({},{'\_id':0,'cates':1}).limit(300):

print(i)

{'cates': ['北京58同城', '北京二手市场', '北京二手家电', '北京二手冰柜']}

{'cates': ['北京58同城', '北京二手市场', '北京二手家电', '北京二手洗衣机']}

{'cates': ['北京58同城', '北京二手市场', '北京二手家电', '北京二手冰箱']}

{'cates': ['北京58同城', '北京二手市场', '北京二手台式机/配件']}

{'cates': ['北京58同城', '北京二手市场', '北京二手笔记本']}

{'cates': ['北京58同城', '北京二手市场', '北京二手家电', '北京二手冰柜']}

{'cates': ['北京58同城', '北京二手市场', '北京二手台式机/配件']}

pipeline2 = [

{'$match':{'$and':[{'pub\_date':'2015.12.25'},{'time':3}]}},

{'$group':{'\_id':{'$slice':['$cates',2,1]},'counts':{'$sum':1}}},

{'$sort':{'counts':-1}}

]

看一下pipline2整理出来的数据是什么样的。

**for** i **in** item\_info.aggregate(pipeline2): print(i)

{'\_id': ['北京二手图书/音像/软件'], 'counts': 30}

{'\_id': ['北京二手家电'], 'counts': 26}

{'\_id': ['北京二手服装/鞋帽/箱包'], 'counts': 26}

{'\_id': ['北京二手文体/户外/乐器'], 'counts': 19}

{'\_id': ['北京二手母婴/儿童用品'], 'counts': 17}

最后要定义一个生成器，连续不断的生成数据。yield [i['\_id'][0],i['counts']]每一组数据都会被装进一个列表中，列表中前面的\_id就是名称即类目，因为它也是一个列表，故取它的第1个元素。后面是它的计数。

def data\_gen(date,time):

pipeline = [

{'$match':{'$and':[{'pub\_date':date},{'time':time}]}},

{'$group':{'\_id':{'$slice':['$cates',2,1]},'counts':{'$sum':1}}},

{'$sort':{'counts':-1}}

]

for i in item\_info.aggregate(pipeline):

yield [i['\_id'][0],i['counts']]

**for** i **in** data\_gen('2016.01.10',1):

print(i)

['北京二手家电', 109]

['北京二手母婴/儿童用品', 62]

['北京二手文体/户外/乐器', 49]

['北京二手服装/鞋帽/箱包', 44]

['北京二手数码产品', 44]

['北京二手家具', 42]

['北京二手办公用品/设备', 32]

['北京二手图书/音像/软件', 26]

['北京二手台式机/配件', 24]

进行作图可视化，options直接从highchart的官网复制得来。

options = {

'chart' : {'zoomType':'xy'},

'title' : {'text': '发帖量统计'},

'subtitle': {'text': '2016.01.10二手物品在随后7天内，交易时长为1天的类目分布占比'},

}

series = [{

'type': 'pie',

'name': 'pie charts',

'data':[i **for** i **in** data\_gen('2016.01.10',1)]

}]

charts.plot(series,options=options,show='inline')

与上面的图表的分布类似，即发帖量越多的类目，在一天内被交易出去的数目也多。但这也只能定性的大概看一下，想要看两个变量之间的相关情况，可以用散列图。

在1天内，2天内，3天内被交易出去的数量占比。2天内的要把第1天的加上，3天内的要把前2天的加上，如此类推。

可以再设计一个，在一天内被交易出去的数量占总发帖量的比例的排序。

本节回顾：

这节课我们使用pipeline方法统计了类目分布占比。

管道模型就像是一层层的管子，从数据库中通过管道取出数据。

这些过滤管道，第一个都是字典的结构，我们先学习最主要的前四个。

$match

$group

$sort

$limit

现在你已经可以进行常见的数据统计了，来挑战本周的大作业吧。

## 第四节练习项目 绘制热销商品的分布饼图。

bj.ganji.com/wu

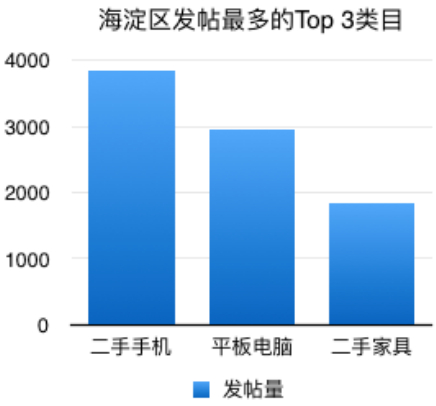
统计一天内交易成功的商品中，它们的城区分布是怎样的，然后绘制一天内交易成功的商品中，各城区交易量的饼图。

看一下哪些类目的商品是热销商品。发帖量越多，被交易出去的越快，交易出的商品对比总发帖量占比高的类目即热销商品。每隔两个小时采集一下数据，检测是否被交易出去。

## 第三周实战作业 分析二手商品行情。

根据赶集的北京二手商品源数据，统计两个数据项，在jupyter notebook里面绘制图表，并把结果导出到csv。

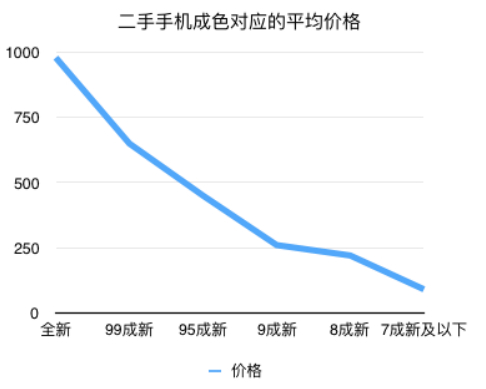
数据项1



以上为示例图，朝阳，通州等其他区以此类推，每个城区都找出top3的大类目和对应的发帖量。

1、北京市各个城区，发帖量最多的3个大类目分别是哪些，发帖量有多少。时间范围自定，数据量越大统计结果越趋于准确。

数据项2



以上为示例图，其他的大类目同样以此类推，没有成色字段的类目不做统计。

2、各个大类目中，成色情况对应的平均价格分别是多少。时间范围自定，数据量越大统计结果越趋于准确。

## 第三周实战作业讲解

**问题1：某段时间内，北京各个城区发帖数量的 top3 类目**

**已知：**

1. 某个时间段，例：2015.12.25~2015.12.27
2. 北京各个城区，例：朝阳

**求解：**

1. 发帖数量 top3

也可以拆解成2步，第1步，全部的发帖数量，第2步，找发帖数量的top3

x轴是对应的各个城区，y轴是发帖数量

**所需数据结构：**

1. 原始 :series = [{'name': 'name','data': [100]},{'name': 'name','data': [100]}]

每一次都要往函数中传入一个数列，函数才会生成我们想要的数据结构。原始结构是由一个个的字典组成的列表，每个字典对应于生成的图表中的一个个柱。每个字典中又有两组字段，名称和数据，分别对应于x轴和y轴。

1. 实际上：{name:类目，data:发帖量}
2. 目标：{'\_id': ['北京二手家电'], 'counts': 175}

示例：

series = [

{

'name': 'name',

'data': [100],

'type': 'column'

},{

'name': 'name2',

'data': [102],

'type': 'column'

}]

options = {

'chart' : {'zoomType':'xy'},

'title' : {'text': '发帖数量最大的类目'},

'subtitle': {'text': '数据图表'},

'yAxis' : {'title': {'text': '数量'}}

}

charts.plot(series,options=options,show='inline')

使用管道模型来对数据进行筛选

第一步，构建管道

某个时间段和北京各个城区，首先是match的过滤管道，match要符合2个条件。首先是pub\_date这个字段，要求是一个时间段，故要传入2个参数，使用gte和lte来实现。第二个筛选条件是各个城区，筛选的是area字段。area字段的结构是一个字典里嵌套了一个列表，{'area':[1,2,3]}，没有办法去直接描述它，去进行一个集合的筛选，只要它里面包含我们想找的，那么它就是匹配的，用$all，包含操作符来实现，可以写一个多值的列表结构。

目标是要把数据整理成{'\_id': ['北京二手家电'], 'counts': 175}的形式，只需要两个字段，使用group来进行过滤，一是把名称换进去，二是使用group中sum的操作符来进行运算的统计。先是id的字段，是各个城区top3的类目，也就是课程中所讲的15个大类，同样用slice分片的方法就可以实现。第二就是求和，用counts字段，每找到1个数量就加1。

因为需要top3，所以把limit设定为3即可。

用for循环来打印看结果。

pipeline1 = [

{'$match':{'$and':[{'pub\_date':{'$gte':'2015.12.25','$lte':'2015.12.27'}},{'area':{'$all':['朝阳']}}]}},

{'$group':{'\_id':{'$slice':['$cates',2,1]},'counts':{'$sum':1}}},

{'$limit':3}

]

**for** i **in** item\_info.aggregate(pipeline1):

print(i)

{'\_id': ['北京二手图书/音像/软件'], 'counts': 32}

{'\_id': ['北京其他二手物品'], 'counts': 26}

{'\_id': ['北京二手办公用品/设备'], 'counts': 86}

然后就是去定义一个函数，来生成数据。函数的核心就是pipeline管道和一个for循环。把管道中的日期和地区换为要传入的参数。

def data\_gen(date1,date2,area,limit):

pipeline1 = [

{'$match':{'$and':[{'pub\_date':{'$gte':date1,'$lte':date2}},{'area':{'$all':area}}]}},

{'$group':{'\_id':{'$slice':['$cates',2,1]},'counts':{'$sum':1}}},

{'$limit':limit},

{'$sort':{'counts':-1}}

]

for i in item\_info.aggregate(pipeline1):

data = {

'name': i['\_id'][0],

'data': [i['counts']],

'type': 'column'

}

yield data

调用函数进行测试

**for** i **in** data\_gen('2015.12.25','2015.12.27',['朝阳'],3):

print(i)

{'name': '北京二手图书/音像/软件', 'type': 'column', 'data': [32]}

{'name': '北京其他二手物品', 'type': 'column', 'data': [26]}

{'name': '北京二手办公用品/设备', 'type': 'column', 'data': [86]}

把数据传入到highchart固定的格式中

series = [i **for** i **in** data\_gen('2015.12.25','2015.12.27',['朝阳'],5)]

options = {

'chart' : {'zoomType':'xy'},

'title' : {'text': '发帖数量最大的类目'},

'subtitle': {'text': '数据图表'},

'yAxis' : {'title': {'text': '数量'}}

}

charts.plot(series,options=options,show='inline')

**问题2：某段时间内各大类目中成色对应的平均价格**

**已知：**

1. 某个时间段，例：2015.12.25~2015.12.27
2. 各大类目，例：北京二手手机
3. 单个物品价格，例：某品牌二手电视，价格1000元

想要得到平均价格，必须要知道单个物品的价格

**求解：**

1. 平均价格

在group中，用求平均数的方法来求得平均价格。

**所需数据结构：**

最终的目标是从数据中筛选出价格和成色对应的名称。但是实际上得到的是一个包含多个价格的列表，列表中的价格是与成色一一对应的，成色已经写在option的x轴中了。不会是像问题1中那样得到name,date,type一一对应的结构。这里还会用到一个排序的方法。

1. 原始：series = ['data1'，'data2'，'data3']
2. 实际上：[全新对应的价格，9成新对应的价格，8成新对应的价格，7成新对应的价格]
3. 目标：{'avg\_price': 300.0, '\_id': '7成新及以下'}

例子：

把data传入到charts.plot函数中，用不同的成色作为x轴，y轴则是价格。

作出来的图中并没有日期，可以预计，输入的时间越长，得到的曲线可能就会越平滑。

data = [1,2,3,4,5,6,7]

options = {

'title': {'text': '新旧-价格'},

'xAxis' : {'categories': ['报废机/尸体','7成新及以下','8成新','9成新','95成新','99成新', '全新']},

'yAxis' : {'title': {'text': '价格'}},

}

charts.plot(data,show='inline', options=options)

for i in item\_info.find({},{'\_id':0,'look':1}).limit(100):

print(i)

在所得到的商品数据中，有些商品是没有新旧的说明的，即没有look的字段，在定义pipeline的时候要进行排除。即要排除look为‘-‘的商品。使用$nin，即not in来进行排除。它和上面的$all操作符的原理是一样的。all如果写2个，表示2个都有的条件。nin则表示不在这里面，反向排除。{'look':{'$nin':['-']}}表示只要look中包含’-’，都会被排除。写完pipline2后发现层级关系很混乱，可以在jsbeautifier.org这个网站上把代码转化为更加清晰的结构。

group是以成色进行重组，统计不同成色的平均价格。'\_id':'$look'，以成色进行分组。{'$avg':'$price'}是对价格求平均值。综合起来就是求出每种成色下面物品的平均价格。

{'$sort':{'avg\_price':-1}}进行排序，在排序的时候要指明是对谁进行的排序，这里是对group中生成的avg\_price字段进行排序。

<https://docs.mongodb.org/v3.0/reference/operator/aggregation/not>

这里可以看到很多求值的操作符。

pipeline2 = [

{'$match':{'$and':[{'pub\_date':{'$gte':'2015.12.25','$lte':'2015.12.27'}},{'cates':{'$all':['北京二手手机']}},{'look':{'$nin':['-']}}]}},

{'$group':{'\_id':'$look','avg\_price':{'$avg':'$price'}}},

{'$sort':{'avg\_price':-1}}

]

用循环来查看结果

**for** i **in** item\_info.aggregate(pipeline2):

print(i)

{'avg\_price': 1849.7894736842106, '\_id': '全新'}

{'avg\_price': 1704.5, '\_id': '99成新'}

{'avg\_price': 1626.923076923077, '\_id': '95成新'}

{'avg\_price': 1187.5151515151515, '\_id': '9成新'}

{'avg\_price': 730.0, '\_id': '8成新'}

{'avg\_price': 300.0, '\_id': '7成新及以下'}

{'avg\_price': 220.0, '\_id': '报废机/尸体'}

看到还有一个问题，用avg求平均之后后面很长的一段小数。在下面定义函数的时候还要进行一次转换。

定义函数

**def** data\_gen2(date1,date2,cates):

pipeline = [

{'$match':{'$and':[{'pub\_date':{'$gte':date1,'$lte':date2}},

{'cates':{'$all':cates}},

{'look':{'$nin':['-']}}

]}},

{'$group':{'\_id':'$look','avg\_price':{'$avg':'$price'}}},

{'$sort':{'avg\_price':1}}

]

**for** i **in** item\_info.aggregate(pipeline):

**yield** i['avg\_price']

调用函数查看结果，从经验上来看，就是越新越贵。

data = [i **for** i **in** data\_gen2('2015.12.24','2016.01.10',['北京二手手机'])]

options = {

'title': {'text': '新旧-价格'},

'xAxis' : {'categories': ['报废机/尸体','7成新及以下','8成新','9成新','95成新','99成新', '全新']},

'yAxis' : {'title': {'text': '价格'}},

}

charts.plot(data,show='inline', options=options)

如何导出 csv?

mongoexport -d database -c collection -o output/path.csv

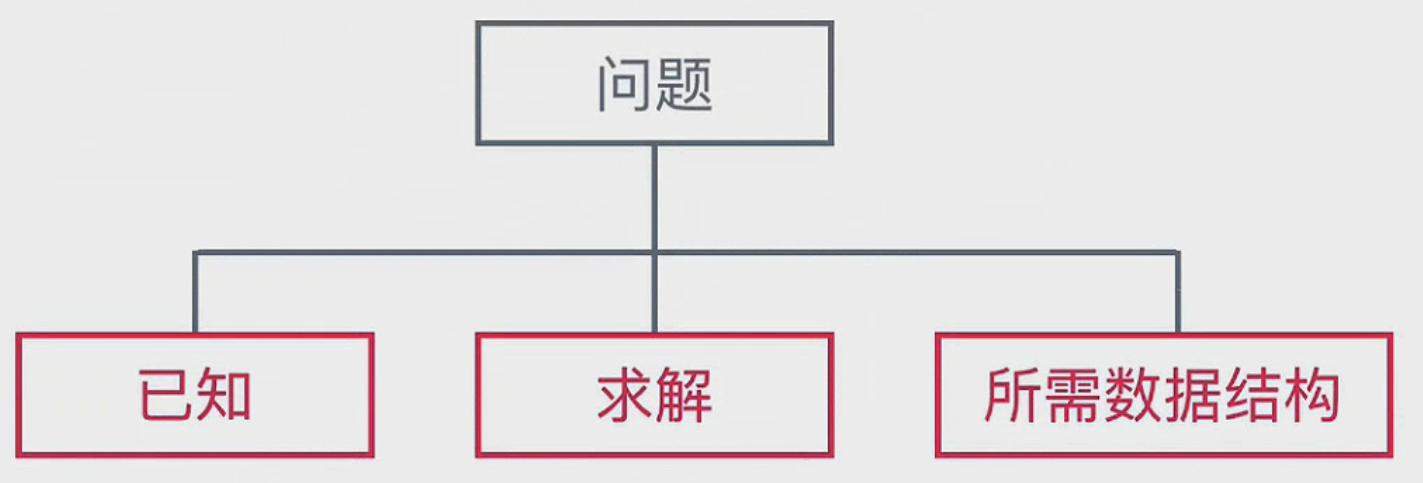
mongoexport -d ceshi -c item\_infoZ -o Users/Hou/Desktop/ddd.csv

mongoexport -d ceshi -c item\_infoZ -o Users/Hou/Desktop/ddd.json

本节回顾：

这节课我们使用pipeline方法做了两个图表

运用了一种解决问题的方法论，让问题更加清晰。



掌握了$match，$group，$sort，$limit的使用方法。

如果把这些展示给其他人呢，做一个网页。

## 第三周补充资料

MongoDB之索引

[https://github.com/qianjiahao/MongoDB/wiki/MongoDB之索引](https://github.com/qianjiahao/MongoDB/wiki/MongoDB%E4%B9%8B%E7%B4%A2%E5%BC%95)

解读：这周最主要的就是MongoDB的操作，这份资料是MongDB索引方面最好的资料，作为课程之外的补充，提供给希望掌握更多索引方法的同学。

## 第四周 用Django搭建网站

恭喜你突破重重挑战，来到了第四周的课程。在这一周里，我们将使用成熟的web框架Django来搭建一个网站，把之前辛苦爬取和统计的数据呈现在上面。

即使你不懂得前端开发也没关系，我们会使用一个漂亮的bootstrap模板，调用里面写好的css样式，去完成这个网站。

学习到这里，想必你会发现，学习一门编程语言，就是在理解这门课文思考的方式。当你掌握了这种思考方式之后，你就有能力去用编程做更多的事情了。

那么，让我们开始吧。

## 课前准备

这周课程需要做好3项准备：

1。安装Django

2. 在虚拟环境中创建Django项目

3. Django的设置与管理

下面开始逐条说明：

1、安装Django

无论是windows,linux还是mac，都可以使用pip来安装。

如果想安装到python3中，需要输入：

pip3 install Django

如果同时安装了python2和python3两种版本，可能会遇到安装目录的问题，可以换成

python3 -m pip install Django

如果想安装到python2中，需要输入：

pip install Django

为了检查是否安装成功，可以在终端里输入：

python3

import django

django.VERSION

会显示当前的Django版本。

2、在虚拟环境中创建Django项目

当我们选择在虚拟环境中创建Django项目时，就能拥有一个干净、独立的开发环境，不必担心项目之间会互相影响。

完成这个步骤的最简单的方法是使用pycharm来创建，具体如下：

1）创建新项目

2）新建虚拟环境

在interpreter右侧的小齿轮点开能看到图中右边的菜单，选择create virtualEnv。

这个功能只有pycharm专业版才有，社区版是没有的，所以或者下载安装专业版，或者使用命令行手动安装。

<https://djangogirlstaipei.gitbooks.io/django-girls-taipei-tutorial/content/django/installation.html>

在弹出框中填写信息，建议name使用‘username\_venv‘的形式，来提醒自己这是虚拟环境。路径和python环境根据个人情况选择就好，最下面的记得勾选，代表着可以在虚拟环境中使用之前安装好的库，填好后点击OK。

3）建立Django项目和Django App

回到这个窗口开始创建项目，在location里面选择一个路径，把原来的untitled1改名为mysite。

点开More Settings，把Application name里面填写上项目的名字：ganji。

点击Create

然后看到pycharm中左导航中的文件结构，就表明创建成功了。

刚才我们是通过pycharm这种简单的方法完成了这个步骤，还有一种方法是使用命令行来创建虚拟环境、Django项目和Django App，可以在下面这份教程中查看详细说明：

<https://djangogirlstaipei.gitbooks.io/django-girls-taipei-tutorial/content/django/installation.html>

3、Django的设置与管理

manage.py是Django提供的命令行工具，可以利用它执行很多工作，使用方法类似这样：

python manage.py <command> [options]

如果你想要了解有哪些命令可以使用，输入help会列出所有：

python manage.py help

1）下面我们来使用manage.py启动服务器：

首先打开pycharm，切换到下面的Terminal，这样能直接打开当前环境和文件的路径。如果是使用命令行或终端的话，需要先切换到虚拟环境，再打开项目所在路径，具体可以查看刚才的教程。

然后在Terminal中输入：

python3 manage.py migrate

python3 manage.py runserver

现在打开浏览器，输入<http://127.0.0.1:8000/>或是<http://localhost:8000/>

会看到你的Django网站已经在web server上成功运行了。

有一件事需要记住：之后 改动了文件，并且想要提交到服务器上时，都需要runserver，也就是输入：

python3 manage.py runserver

2）把Django app加入设置

每一个Django project里面都有多个Django apps，这里说的不是手机app，而是application应用。在实际使用中，我们通常会根据功能分成不同的app，方便未来的维护和重复使用。

比如说，当我们要做一个类似Facebook的网站时，根据功能就有这样的apps:

账号管理——accounts app

好友管理——friends app

时间线管理——timeline app

动态消息管理——news app

而未来，当我们需要写个购物网站，需要账号功能的时候，accounts app就可以被重复使用。

之前我们用pycharm建立 了ganji这个app，但为了让Django知道要管理哪些apps，还需要调整设置。

在pycharm中打开mysite/setting.py，找到INSTALLED\_APPS，把ganji增加在最下面：



3）设置Django管理后台

Django诞生于新闻网站的环境中，所以很重视内容管理，提供了管理后台，让使用者方便新增或修改网站内容。

这个管理后台，在Django中以内置app的形式存在，叫做：Django Admin。现在我们来进行设置：

1. 建立管理员账号

在pycharm的terminal里面，输入：

python3 manage.py createsuperuser

会弹出提示让你分别输入用户名、邮件、密码，要牢记这些。然后就完成了superuser的账号设置了。

2. 使用管理后台

执行runserver指令，然后进入<http://127.0.0.1:8000/admin>，可以看到管理后台的登录页面。

输入刚才创建superuser的账号密码，进入管理后台。

会看到这样的后台，祝贺你搭建好了原始版的Django，更多内容我们在课程中讲解。

注：手册可能无法穷举所有的问题，其他的安装问题或使用说明可以查看官网：

Django官网：<https://www.djangoproject.com>

## 第一节课程 使用Django的MTV模型搭建网站

第1步 理解MTV模型

Request DjangoWeb (Views Models Templates) Response

第2步 创建Django虚拟环境

新建项--->Django--->Create VirtualEnv--->Name Location, check ‘Inherit global site-packages’ --->项目location--->Create

第3步 在templates中创建我们的网站

要使用manage.py来创建，因为这个文件帮我们设置了很多环境变量。

为什么是一个app呢，我们创建的网站，博客，新闻站点等，django都把它们认为是一个application

**创建django的站点**

点最下面的terminal

python3 manage.py startapp django\_web

项目中已经有一个名为django\_web的站点了。

用MTV模型的观点来查看站点文件

templates是负责页面可视化的文件夹。

views.py是负责取数据和查找templates的文件。

models.py是负责从数据库中取数据的文件

urls.py是负责分配urls的文件

**在setting中以app的方法添加我们的站点**

Django\_sample--->settings.py是django站点的一些默认设置和安全设置。在其中找到INSTALLED\_APPS，把刚才创建的站点添加上去，这样Django就能够识别了。

在最后面加上一句：

‘django\_web‘

接下来我们倒着去做。

正常的顺序是：发送一个request，第1步经过url，第2步，url找到了对应的view，第3步，view去从model和template中去找对应的数据库和网页。

如果没有数据库和网页，有链接和view，也是不能够访问到任何数据的。所以要从最底层的网页和数据库开始进行设计。然后再添加调用它的视图函数，再给它分配一个链接。因为今天的链接都是静态的，所以model层是用不上的。

今天能用到views, templates, urls

**添加可视化的模板**

在templates文件夹中添加一个名为index的html file，把第一周中用过的网页复制过来。

**定义templates的视图函数view.py。**

from django.shortcuts import render

def index(request):

return render(request, 'index.html')

函数的名称可以任意定义。

传入的参数是抽象意义上的request函数，这个request也许是点击，也许是请求等。

view视图传入request之后，用render函数来调用index.html页面，再用return来返回这个index.html文件。render函数必须要有这个request参数。

**分配url**

在views.py中只是定义了一个函数，一个网页是否能够出现在网站上并且被客户端访问，网址是唯一的途径，必须要在urls.py中给它分配一个网址，调用这个函数，

**from** django.conf.urls **import** url  
**from** django.contrib **import** admin  
urlpatterns = [  
 url(**r'^admin/'**, admin.site.urls),  
 url(**r'^index/'**, admin.site.urls),  
]

使用正则表达式，表示所有以index开关的文件都会指向后面admin.site.urls这个站。

为什么要使用正则呢，经常会遇到像?page=1这个的页面，对于django网站来说，不能说每有一个页面就去写一个视图文件，所以就使用正则表达式来进行模糊匹配，把多个链接指向一个视图函数。

urlpatterns是一个列表，中间装着两个url函数，但必须先import函数，才能够引用它。

**from** django.conf.urls **import** url  
**from** django.contrib **import** admin  
**from** django\_web.views **import** index  
  
urlpatterns = [  
 url(**r'^admin/'**, admin.site.urls),  
 url(**r'^index/'**, index),  
]

注意是从django\_web这个文件夹，也即创建的站点文件夹中引入的index函数，不要引用项目的文件夹。总体意思就是从站点文件夹的views中引入了一个index函数/视图，给它分配了一个链接，恰好也是使用index这个名字。

**运行查看结果**

python3 manage.py runserver

出现的地址是

<http://127.0.0.1:8000/>

因为我们给它分配的链接是index，所以要访问下面的地址才能得到网页。

<http://127.0.0.1:8000/index/>

**复制网站相关文件**

但所有的样式都不存在，要把之前制作的网站的相关文件static也复制过来，其中包括样式和图片。把它放在站点项目文件夹下，保证static与templates和django\_web这三个文件夹是平级的关系。

**使用Django的模板语言调用样式**

然后在index.html文件中引用样式。Django有一个叫template tag模板语言，类似于以前使用的字符串格式化函数{}{}{}.formate(1,2,3)

把下面这句写在index.html的开头。意思是获取了所有static的路径。

{% load static %}

下面再调用static中的样式时，要写为

href = “{% static ‘new\_blah.css’ %}”

**在setting.py中添加static的地址。**

找到STATIC\_URL，表示相对目录。在后面添加一句。不用担心文件的绝对路径在哪里，setting.py中有一句BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))，就是使用一个函数使用相对路径找到绝对路径的方法。无论放在哪里，系统都能够找到。在绝对路径BASE\_DIR后面加上static的相对路径，就找到了static这个文件夹。

STATIC\_URL = **'/static/'**STATICFILES\_DIRS = (os.path.join(BASE\_DIR, **'static'**),)

**在模板中替换css和image的地址**

因为是把new\_blah.css样式放在了css文件夹下，还要把index.html中引用样式的地址改为

href = “{% static ‘css/new\_blah.css’ %}”

同样，把所有的images替换为{% static ‘images

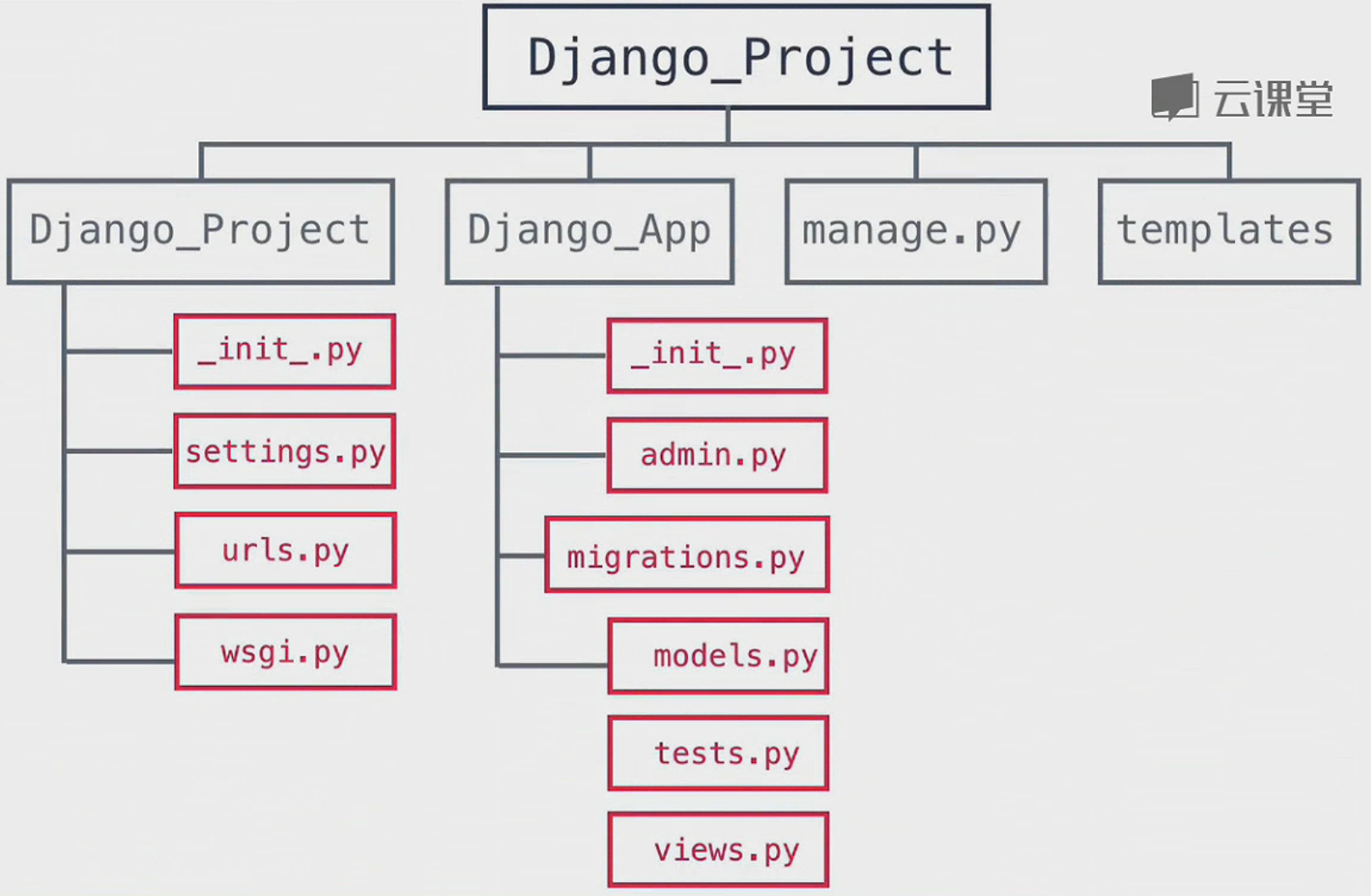
把所有的.jpg替换为.jpg’ %}

把所有的.png替换为.png’ %}

本节回顾

这节课我们使用django框架重新实现了一次之前的网页

最重要的就是理解Django的MTV模型



实际上，Django还有一个很重要的知识点，就是模板语言

下节课学习如何使用模板语言进行数据分布展示，即做分布符

## 第一节练习项目 搭建个人博客

把4.1源代码中的pure-layout-blog这个网站，用django搭建出来。

## 第二节课程 熟悉Django的模板语言

使用模板语言进行数据分页展示

第一步 理解上下文

理解Django中上下文的机制

第二步 使用模板语言

使用Django中的模板语言实现对网页结构的控制

第三步 制作分页器

第一步 理解上下文

理解Django中上下文的机制

render()函数，即渲染，render(request, x.html, context)，用来接收3个参数。第1个参数是接收从url传来的用户进行操作的request，第2个参数指定要返回哪个模板，即要指定template的名称，实际上就是网页，第3个参数为context，上下文。如何理解，如有一个填空，There’s a Block waiting in the sky. 空格中填入不同的单词，这句话就可以表达不同的意思。这种机制被用在Django中，通过替换不同的词来进行不同句意的表达。可以把要填写的句子理解成我们的网页，网页的内容存在我们的数据库中，context参数就是把数据库中的数据和网页中要加载的内容进行一个映射，把我们要展示的参数替换到上面。类似于在句子中填入不同的单词。

本节课的目标就是要做的是加一个分页符，点一下就可以切换不同的页面，加载不同的数据。

**第1步 在views.py中构建context**

在context中写出要替换的这3个部分，并把context传递到render函数中。context的参数就是一个字典的结构，在context中写入我们想要传入到index中替换具体某个位置的参数。

<**h3**><**a href="#"**>The blah</**a**></**h3**>

<**p class="description"**>Just say something.</**p**>

<**span class="rate-score"**>4.5</**span**>

查看上节课中的网页，在实际中网页中加载不同的内容时大的框架是不变的，改变的只是title, description和rate，现在要替换这三个部分的内容。现在就在context中写入要替换的内容。并把context作为参数传入到render函数中。

**from** django.shortcuts **import** render  
**def** index(request):  
 context = {  
 **'title'**: **'Just a title'**,  
 **'des'**: **'Just a description'**,  
 **'scores'**: **'1.0'**,  
 }  
 **return** render(request, **'index.html'**,context)

要传递的内容已经构建出来了，但模板template怎么才能知道要替换的内容就是这三部分呢。即填空的空怎么去写呢。

**第2步 改写模板template，也即网页。**

与上一节中load static的用法类似，替换第一个li中的内容

<**h3**><**a href="#"**>{{ **title** }}</**a**></**h3**>

<**p class="description"**>{{ **des** }}</**p**>

<**span class="rate-score"**>{{ **scores** }}</**span**>

python3 manage.py runserver

刷新网页，网页的内容就已经更新了。

**第3步 使用model层，即使用数据库**

上面实际上是模拟了一个从数据库中提取出来的一部分数据。如何直接从数据库中提取数据并传递到网页中呢。就要使用model层了。model层相当于数据库的代理，它从数据库中取出数据来，放在上下文中，再映射到模板层中，就能实现同一个页面的样式加载不同的数据的功能。

django实际上是自带数据库的，但我们要使用MongoDB数据库。要想使用MongoDB数据库，需要在setting中进行设置。之所以要在setting中引入数据库，是因为在运行的过程中django先从setting中读取复杂的变量及设置，进行加载，然后把加载的数据沿用到model层和view层中。

要在setting.py最后加上两句，导入mongodb数据库。

**from** mongoengine **import** connect  
connect(**'wbsite'**, host = **'127.0.0.1'**, port=27017)

然后在model层中引入mongodb，因为现在model层没有任何其它的内容，暂时不会引起名称的冲突，所以可以使用\*来引入所有的包。为什么要使用mongoengine而不使用pymongo呢，pymongo是mongoDB开发的python插件，它的操作方式比较底层，在第2周的课程中使用了非常复杂的方法进行了数据库的索引，它的使用方法不够人性化，不够python化，而使用mongoengine操作，则可以把数据库当作对象来操作。

使用mongoengine来连接以前的数据库，建立数据模型，即model层的功能，必须要定义一个类

**from** django.db **import** models  
**from** mongoengine **import** \*  
*#以下两句只在测试models.py时有用。***from** mongoengine **import** connect  
connect(**'wbsite'**, host = **'127.0.0.1'**, port=27017)  
*#类要使用大驼峰来命令，Document不是参数，而是继承的类的名称，即这个ArttiInfo的类是继承于mongodb中的Document类。  
#ArtiInfo这个类就定义了我们的数据结构是什么样的。***class** ArtiInfo(Document):  
 *#定义信息的key, 叫des, title, score, 这是对象的名称des, title, score必须要严格上与数据库中的key的名称相同。StringField即字符串属性。* des = StringField()  
 title = StringField()  
 scores = StringField()  
 *#tag是一个列表，列表中的每一个元素又是一个字符串，这样套着写就行了。* tags = ListField(StringField())  
  
 *# 由于MongoDB中存在不同的collections，还要把使用的collection告诉models  
 #meta仅限于在数据库已经存在的状态下使用* meta = {**'collection'**: **'arti\_info3'**}  
  
*#单独运行时会出错，因为作为项目运行时，django要先从setting中读取设置的数据，单独运行此文件时没有import所需要的包，可以先把setting中的最后的两行导入数据库的语句 复制过来进行测试。*

*#查看ArtiInfo中的数据，它是mongoengine的对象。它使用的完全是python中面向对象的方法。i是每一个document，相当于是一个个的文件，它是一个字典的结构，想要取出来它的des的值，就可以用print(i.des)的方法。***for** i **in** ArtiInfo.objects:  
 print(i)  
  
*#打印出来数据结构，就是存在于arti\_info3中的数据，用了mongoengine中的方法对构造的这个类使用了object方法，使它变成了可以不断生成数据的东西。i是每一个内容，对每一个内容使用定义好的方法。这种使用对象的方法比之前用的find函数要快很多。注意在mongoengine中变量的名称和数据库中key的名称必须要严格的一一对应。***for** i **in** ArtiInfo.objects:  
 print(i.title, i.des, i.scores, i.tags)

*#由于数据较多，可以使用以下方法限定数据个数, 是一个分片功能。相当于pymongo中的limit功能。*

**for** i **in** ArtiInfo.objects[:1]:  
 print(i.title, i.des, i.scores, i.tags)

**第4步 在view层中引用数据库**

model层定义结束后就可以在view层中引入真正的数据库中的数据了。

**from** django.shortcuts **import** render  
*#从model中引入函数***from** django\_web.models **import** ArtiInfo  
**def** index(request):  
 *#把类进行一个实例化，可以理解为把右边类中产生的数据全部放在左边的变量中了。* arti\_info = ArtiInfo.objects[:1]  
 *#把context中的内容替换为数据库中的内容，只有一个数据* context = {  
 **'title'**: arti\_info[0].title,  
 **'des'**: arti\_info[0].des,  
 **'score'**: arti\_info[0].scores,  
 }  
 **return** render(request, **'index.html'**, context)

刷新网页，网页中的第一个内容就替换为数据库中的内容了。

context参数的作用就是把数据库中的数据放在render函数中作为一个参数，把数据和模块放在一起进行渲染，返回一个渲染后的网页。

**第5步 使用结构化控制的方法在view层中加载数据库中的数据**

数据库中的数据可能非常多，比如说有1000条，要用一个更高效的方法进行网页数据的加载，第1步是把1000条数据变成多少页，每页显示多少条。比如每页加载4条。展示数据用一些结构控制的方法来实现。

**1。首先引入Paginator**

**from** django\_web.models **import** ArtiInfo  
*#引入一个函数，作用就是把非常多的数据进行分页***from** django.core.paginator **import** Paginator

**2。Paginator基本操作**

在交互的环境下比较方便展示。

打开python Console

**from** django.core.paginator **import** Paginator

#构造一个连续的序列，列表字符串都可以。

iter = ‘abcdefghijklmn’

#使用Paginator构建一个分页器，也就是把它实例化，它的第一个参数就是要被分页的内容，必须是一个连续的内容。第二个参数是多少个元素放在一页上。

paginator = Paginator(iter, 4)

#对分页器使用page方法，得到的就是第1页。得到的还是一个paginator的对象，用page1这个变量来指向第1页这个对象。

paginator.page(1)

page1= paginator.page(1)

#对page1这个对象使用方法。用object\_list方法看第1页中的内容。

page1.object\_list

page3 = paginator.page(3)

page3.object\_list

page3.has\_next() #看第3页后有没有下一页

page3.number #获得page3当前的页码

page3.paginator.num\_pages #获得总共有多少页

paginator.num\_pages 同上

page4 = pageinator.page(4)

page4.has\_next() #False

**3。在view层中构造分页函数**

下面就使用分页的方法把数据库中的内容加入到我们的分页中。数据库中几千条数据都会按页分为每页4个数据。

**from** django.shortcuts **import** render  
*#从model中引入函数***from** django\_web.models **import** ArtiInfo  
*#引入一个函数，作用就是把非常多的数据进行分页***from** django.core.paginator **import** Paginator  
  
*# Create your views here.***def** index(request):

*#每页显示的数据个数* limit = 4  
 *#把类进行一个实例化，可以理解为把右边类中产生的数据全部放在左边的变量中了。* arti\_info = ArtiInfo.objects[:4]  
 paginator = Paginator(arti\_info, limit)  
 *#* request.*GET.get()的作用就是把函数传入的request参数变成了一个页码，并用一个page变量来保存这个页码* page = request.GET.get(**'page'**,1)  
 *#把页码输入到paginator.page中就得到了本页, 把本页的数据装入到context中就可以传入到model中了。相当于把我们想要的数据打了一下包，每一页有4个数据，找到数据的title, des, score就可以完成我们的填空了。* loaded = paginator.page(page)  
  
 *#把context中的内容替换为上面得到的打包的数据  
 #template模板只认前面的key，和后面写的内容没有关系。* context = {  
 **'ArtiInfo'**: loaded  
 }  
 **return** render(request, **'index.html'**,context)

**第6步 使用循环在template层中批量生成内容**

在model中使用一个for循环打印出来了想要的数据，在template层中也应该使用类似的循环达到同样的效果，这样，只用一个li的结构就能够实现原来页面上所有的内容。把index.html的内容修改为如下。

<**ul class="articles"**>  
 {% **for item in ArtiInfo** %}  
 <**li**>  
 <**img src="**{% **static 'images/0001.jpg'** %}**" width="100" height="91"**>  
 <**div class="article-info"**>  
 <**h3**><**a href="#"**>{{ **item**.**title** }}</**a**></**h3**>  
 *<!--tag标签是多个的，要再套一个循环-->* <**p class="meta-info"**>  
 {% **for tag in item**.**tags** %}  
 <**span class="meta-cate"**>{{ **tag** }}</**span**>  
 {% **endfor** %}  
 </**p**>  
 <**p class="description"**>{{ **item**.**des** }}</**p**>  
 </**div**>  
 <**div class="rate"**>  
 <**span class="rate-score"**>{{ **item**.**scores** }}</**span**>  
 </**div**>  
 </**li**>  
 {% **endfor** %}  
</**ul**>

python3 manage.py runserver

重新刷新网页，网页的内容就用数据库的内容重新加载了。可以在views中修改每页显示的数量和总的数量。limit = 10, 每页显示10个内容, arti\_info = ArtiInfo.objects[:20], 加载数据库中的20条记录.

**第7步 实现翻页的功能**

以上只是实现了第一部分，通过模板语言的循环控制和传入的context进行了列表页的数据加载，还要实现翻页的功能。

在<ul></ul>后面加一个和ul结构平级的内容。实现网页中觉的类似于<pre 1 of 2 Next>这样的翻页结构。

<**div class="main-content-pagitor"**>  
 <**a**>< Pre</**a**>  
 <**span**> 1 of 2 </**span**>  
 <**a**>Next ></**a**>  
</**div**>

这样就在形式上实现了翻页的功能。

刷新网页查看效果。

回到views中，

**from** django.shortcuts **import** render  
*#从model中引入函数***from** django\_web.models **import** ArtiInfo  
*#引入一个函数，作用就是把非常多的数据进行分页***from** django.core.paginator **import** Paginator  
  
*# Create your views here.***def** index(request):

*#每页显示的数据个数* limit = 4  
 *#把类进行一个实例化，可以理解为把右边类中产生的数据全部放在左边的变量中了。* arti\_info = ArtiInfo.objects[:4]  
 paginator = Paginator(arti\_info, limit)  
 *#* request.*GET.get()的作用就是把函数传入的request参数变成了一个页码，并用一个page变量来保存这个页码*page = request.GET.get(**'page'**,1)

*#*对于page = request.GET.get(‘page’,1)这个语句，request参数是从urls.py中传递过来的。打印request和request.GET来查看request的内容。

print(request)

print(request.GET)

loaded = paginator.page(page)  
  
context = {  
 **'ArtiInfo'**: loaded  
 }  
 **return** render(request, **'index.html'**, context)

打印得出的内容为

<WSGIRequest: GET ‘/index/’>

<QueryDict: {}>

“GET /index/ HTTP/1.1” 200 3988

“GET /static/css/new\_blah2.css HTTP/1.1” 304 0

“GET /static/images/blah.png HTTP/1.1” 304 0

“GET /static/images/0001.jpg HTTP/1.1” 304 0

request对象是<WSGIRequest: GET ‘/index/’>，即请求的/index/中的内容

request.GET是一个类似于字典结构的方法。page = request.GET.get('page',1)，传入两个参数，第1个是后缀的名称，是‘page’，第2个是页数，一共有多少页。在写完下面有页码的分布之后，才能更好的理解这个函数。

回到模板中，写模板层的逻辑控制。

先写index中的下一页的功能，如果后面还有页码的话就可以点击并在点击后跳转到下一页去。要做一个if的判断。

<**div class="main-content-pagitor"**>  
 <**a**><Pre</**a**>  
 <**span**> 1 of 2</**span**>  
 {% **if ArtiInfo**.**has\_next** %}  
 <**a href = "?page=**{{ **ArtiInfo**.**next\_page\_number** }}**"**>Next ></**a**>  
 {% **endif** %}  
</**div**>

刷新网页查看改变，点击下一页跳转到第2页，第3页等，就能看到网页内容的变化。

切换到pycharm中，看到views中的打印的request内容的变化。

<WSGIRequest: GET ‘/index/?page=2’>

<QueryDict: {‘page’: [‘2’]}>

“GET /index/?page=2 HTTP/1.1” 200 3962

<WSGIRequest: GET ‘/index/?page=3’>

<QueryDict: {‘page’: [‘3’]}>

“GET /index/?page=3 HTTP/1.1” 200 3962

<WSGIRequest: GET ‘/index/?page=4’>

<QueryDict: {‘page’: [‘4’]}>

“GET /index/?page=4 HTTP/1.1” 200 3962

request类似于以url链接的形式传过来，在网页中点检查，在sources中查看Doc，点下一页，就能看到?page=2之类的页码。点一下左边的?page=2就能看到Request和Response的内容。在Query String Parameters中就能看到page:2，相当于我们通过‘?’传给服务器的参数的变量名称为page，传给服务器的对应的量为2，3，4等。

然后再实现template中Previous部分和页码显示部分。

<**div class="main-content-pagitor"**>  
 {% **if ArtiInfo**.**has\_previous** %}  
 <**a href = "?page=**{{ **ArtiInfo**.**previous\_page\_number** }}**"**>< **Pre** </**a**>  
 {% **endif** %}  
 <**span**> {{ **ArtiInfo**.**number** }} of {{ **ArtiInfo**.**paginator**.**num\_pages** }}</**span**>  
 {% **if ArtiInfo**.**has\_next** %}  
 <**a href = "?page=**{{ **ArtiInfo**.**next\_page\_number** }}**"**>Next ></**a**>  
 {% **endif** %}  
</**div**>

本节回顾

这节课我们使用了模板语言，对数据进行了分页展示。

context There’s a Starman waiting in the sky

Django中的上下文，其实很像完形填空。

{% if ArtiInfo.has\_next %}

<a>Next</a>

{% endif %}

模板语言的特点，想必你已经印象深刻了。

你可能会说，这个网页不够好看，为了做出一个好看的网页，我们将学习一个UI框架。

下节课教你使用Semantic UI框架，使用semantic UI制作模板。

## 第二节练习项目 给博客增加分布功能

把上一节课的作业中做的网站，增加分页的功能，并且使用58爬取的数据作为数据源。

## 第三节课程 开始使用semantic框架

<https://semantic-ui.com>

<http://www.semantic-ui.cn/>

<http://www.semantic-ui-cn.com/>

semantic是一个用于前端开发的框架，包含前端的样式css部分和一些页面的效果。

它最大的亮点是语义化。用少量的代码就能达到满意的效果。包含了平时开发中应用到的元素。

不需要记住太多，只要参照着官网上的例子来做就行了。

本节的内容包括：

第一个是学会使用container容器来装载网页的一部分内容。

第二个是在网页中制作一个分页，即上一节课中的内容。样式会比上一节中漂亮。

第三个是制作网站的导航。

第四个是网站主体content的部分，把文章的信息以semantic中的content的方式呈现

第五个是做一个可以隐藏的侧栏，现在很多网站都是分栏式的，两栏或三栏式的。

**在head部分中引用css和js的样式**

引入css样式

**<link** rel="stylesheet" type="text/css" href="css/semantic.css"**>**

引入js效果

**<script** src="js/semantic.js"**></script>**

**<script** src="js/jquery.js"**></script>**

**使用semantic UI中的效果**

引入第一个样式container，文字，文章，页面的结构都可以装入到container的结构中。

<http://www.semantic-ui.cn/elements/container.html>

**<div** class="ui container"**>**

**<h3>**A container!**</h3>**

**<p>**Say something**</p>**

**</div>**

**但是现在没有实例中的阴影效果，要加上segment**

**<div** class="ui container segment"**>**

**<h3>**A container!**</h3>**

**<p>**Say something**</p>**

**</div>**

**把页面的内容复制的多一些，查看效果，随着内容的变化，container也随着缩放。**

**<div** class="ui container segment"**>**

**<h3>**A container!**</h3>**

**<p>**Say something Say something Say something Say something Say something Say something Say something Say something Say something Say something Say something Say something Say something Say something Say something **</p>**

**</div>**

上面就了解了如何使用semantic UI中元素的样式发生变化的方法，即不断的给它叠加class类，使它变的更加语义化更具有描述性。

**使用item**

<http://www.semantic-ui.cn/views/item.html#divided>

divided items类可以被看成是一个筐，内容就都放在这个筐中。其中每一部分内容都被称为是item，item中又有图片，还有除图片之外的包含信息，按钮，标签和描述的content部分。content中又包含header，即item部分的主标题，meta，即附加信息，description，即描述。还有extra，使用ui label来形成一个个的小标签。

<**div class="ui text container segment"**>  
 <**div class="ui divided items"**>  
 <**div class="item"**>  
 <**div class="content"**>  
 <**div class="header"**>  
 The title  
 </**div**>  
 <**div class="description"**>  
 <**p**>Say someting</**p**>  
 </**div**>  
 <**div class="extra"**>  
 <**div class="ui label"**>  
 tag1  
 </**div**>  
 <**div class="ui label"**>  
 tag2  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>

复制多个item的项就实现了网页中的多个元素。如果修改ui divided items类为ui items，多个item之间的分割符就会消失，多个item就合并到了一起。从一个样式变化到另一个样式，只需要在class中进行修改，不需要手动添加或删除结构即可实现。

制作分页

<http://www.semantic-ui.cn/collections/menu.html>

<div class="ui pagination menu">

<**div class="ui pagination menu"**>  
 <**a class="active item" href="#"**> < </**a**>  
 <**div class="disable item"**>  
 page 1 of 3  
 </**div**>  
 <**a class="item" href="#"**> > </**a**>  
</**div**>

item并没有一个明显的样式的意义，只是说明这个样式属于上一个样式，二者之间是一个包含关系。item前又能加上很多描述性的说明，来实现不同的效果，active item是处于激活的item，disable item是不能被点击的item。

制作顶栏的菜单

<**div class="ui menu"**>  
 <**div class="header item"**> Menu </**div**>  
 <**div class="item"**> About us </**div**>  
 <**div class="item"**> Location </**div**>  
 <**div class="item"**> Others </**div**>  
</**div**>

header item首字母大写的，会比其它的item大一点。

制作sidebar边栏

<http://www.semantic-ui.cn/modules/sidebar.html>

边栏都是js控制的部分，semantic中的边栏是默认不会出现在网页的可视部分的，除非使用特殊的方法把它固定在页面上。边栏需要有一个pusher的结构。而边栏在被显示的时候，网页的主体部分content是会被push到一边的。被推开的部分就放在pusher中。

<**body**>  
 <**div class="ui sidebar inverted vertical menu"**>  
 <**div class="item"**>option 1</**div**>  
 <**div class="item"**>option 1</**div**>  
 <**div class="item"**>option 1</**div**>  
 </**div**>  
 <**div class="pusher"**>  
 <**div class="ui menu"**></**div**>  
 <**div class="ui text container segment"**></**div**>  
 </**div**>

</**body**>

完成后sidebar是不可见不可操作的，可以用两种方法来使其可见。

第一种方法是加上visible，强行使其显示，这样边栏就一直可见并置于页面的上方。并覆盖一部分页面的内容。

<**body**>  
 <**div class="ui visible sidebar inverted vertical menu"**>  
 <**div class="item"**>option 1</**div**>  
 <**div class="item"**>option 1</**div**>  
 <**div class="item"**>option 1</**div**>  
 </**div**>  
 <**div class="pusher"**>  
 <**div class="ui menu"**></**div**>  
 <**div class="ui text container segment"**></**div**>  
 </**div**>

</**body**>

现在要实现一点击页面中header item中的Menu，边栏就会出现。

给Menu加一个按钮，使其变成可点击的。加上一个i即icon的标签，

<**div class="header item"**>  
 Menu  
 <**i class="content icon"**></**i**>  
</**div**>

刷新网页，Menu右边出现了一个三道横线的结构，如果显示的是小方框，这是内容没有加载出来的原因。打开semantic.css，查找font-face，把src中的绝对路径改为相对路径，把所有的绝对路径改为类似"../fonts/icons.eot"的路径。

现在要对按钮加一些操作，加一些js的效果。

$('.ui.sidebar').sidebar('toggle');

即使用jquery的方法来调用js

在body的最后部分加上js代码。

要使这个Menu的按钮可以点击，第一步就是要告诉这段代码，找到这个Menu的按钮，即定位。$代表查找，$('.header.item')，与css selector的使用方法类似。找到这个元素之后对它使用一个click方法。$('.header.item').click()，与python中面向对象的方法很像，即找到一个对象之后对它使用一个方法或发生一个事件。在这个方法或事件中要调用一个函数，**$**(**'.header.item'**).click(**function** () {})，

这个function中还要调用一个方法.sidebar()方法，$('.header.item').click(function () {.sidebar()})。这个方法在jquery中已经被定义好了。在modules中有一个sidebar方法，即对象使用一个叫作sidebar的函数，使边栏能够移入移出。还要加上一个toggle的参数，即使对象能够滑入滑出。$('.header.item').click(function () {.sidebar('toggle')})

现在还缺少一个对象，使什么对象能够滑入滑出。同样要定位元素$('.ui.sidebar')，完成后的js代码如下，总体的效果是对.header.item的结构使用click方法，点击的时候对.ui.sidebar这个对象使用sidebar函数，完成.ui.sidebar的滑入滑出。

<**script**>  
 **$**(**'.header.item'**).click(**function** () {  
 **$**(**'.ui.sidebar'**).sidebar(**'toggle'**);  
 });  
</**script**>

但点击Menu按钮后没有任何效果。是jquery的问题。

<**script src="js/semantic.js"**></**script**>  
<**script src="js/jquery.js"**></**script**>

有两点错误。一是jquery的名称错误。应该是jquery.min.js。二是script的顺序有对，如果header中同时有js和jquery的代码，要把jquery的代码放在上面才能正确运行。

实际上控制menu的代码还是有小小的问题。对于一个内容很丰富的网页来说，.ui.sidebar这个结构是唯一的，但对于.header.item这个结构很可能不是唯一的。这时候可以对.header.item这个结构定义一个唯一的id结构。id=’menu’，script的代码改为

<**script**>  
 **$**(**'#menu'**).click(**function** () {  
 **$**(**'.ui.sidebar'**).sidebar(**'toggle'**);  
 });  
</**script**>

大的框架已经完成了，余下的工作就是做一些美化的工作。比如说sidebar太宽了，可以做的窄一些。

<**div class="ui thin left sidebar inverted vertical menu"**>

主体内容的居中也并不美观，可以改为默认。把

<**div class="ui text container segment"**>

改为

<**div class="ui container"**>

修改Menu的方法，改为hide，.sidebar(**'hide'**)，这时要修改sidebar，使其默认是打开的，即加上visable，但发现这时只要一点Menu把sidebar隐藏，再点击Menu它也不会再出现了，还要改为‘toggle’

<**div class="ui thin visible left sidebar inverted vertical menu"**>

给header item加上图标，引入了i标签，i标签就是给元素加上icon的。注意要先在semantic.css中font-face的部分把所有icon的路径改为相对路径才能找到。

<http://www.semantic-ui.cn/elements/icon.html>

<**div class="ui thin visible left sidebar inverted vertical menu"**>  
 <**div class="header item"**>  
 <**img src="images/logo2.png"**>  
 </**div**>  
 <**div class="item"**>  
 Document  
 <**i class="browser icon"**></**i**>  
 </**div**>  
 <**div class="item"**>  
 Charts  
 <**i class="bar chart icon"**></**i**>  
 </**div**>  
 <**div class="item"**>  
 Others  
 <**i class="idea icon"**></**i**>  
 </**div**>  
</**div**>

最后一步，把写好的html网页放在Django中，转化为Django的模板。

把html文件放在templates中，并改名为index.html，css, fonts, images, js全部放在static中。

css引用的路径，图片引用的路径，js和jquery引用的路径都要按照第2节中的Django的模板语言方法进行改写。

pusher--->ui container--->ui divided items中，使用循环的方法来自动生成内容。

然后是分页的部分，ui small pagination menu

python manage.py runserver

刷新查看网页。

本节回顾

这节课我们使用了semantic UI框架，设计了这个模板

对照着官网文档，我们完成了菜单、分页、导航、正文、分栏。

很多时候，只需要组合一些元素就能得到我们想要的功能

学会了这么有趣的框架，相信你已经迫不及待的想要把之前的数据放进来了吧

下节课在django站点上制作页面和图表

第三节练习项目 使用Semantic UI框架，实现一个类似Pinterest的网站，效果如图。

需要用到的组件请参考Semantic UI官方文档

<http://www.semantic-ui.cn>

## 第四节 用Django做图表

在Django站点上制作页面和图表

part1 使用Grid进行网站布局

part2 使用jquery生成网页图表

使用jquery调用js使用highchart进行统计图表的生成

part3 使用django模板继承减少重复代码

<http://www.semantic-ui.cn/collections/grid.html#/definition>

使用semantic ui的网格

grid不是网站上的可见元素，它是对网页上的元素和部件的相对位置和距离定位、布局和约束的工具。

使用Equal Width这种风格。如果建立了一个等距的grid，grid中的colum和raw的距离都是一样的，即宽度是相等的。

A grid can automatically resize all elements to split the available width evenly

在第二节课程制作的网页的基础上，删除网页主体部分的内容。

在pusher结构中创建一个新的div结构用来作风格的布局

<**div class="ui equal width grid"**>  
 <**div class="row"**>  
 </**div**>  
 <**div class="row"**>  
 </**div**>  
</**div**>

在创建的ui equal width grid中创建了两行，第一个行中创建两列元素，第二个行中创建一列，用来保存列表部分。

<**div class="ui equal width grid"**>  
 <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui segment"**>  
 One  
 </**div**>  
 </**div**>  
 <**div class="column"**>  
 <**div class="ui segment"**>  
 Two  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui segment"**>  
 List View  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>

查看网页效果。由于定义的是equal width grid，所以第二个行中的一个列的宽度等于第一个行中两个列元素宽度之和。但现在的三个列都比较宽，占整个页面的100%。要调整宽度和第一个行中两个列的颜色。

如何修改宽度，由于是引用的第三方的semantic框架，宽度是在引用的css中已经写好的，如何来修改呢。引入style这种内联样式，就覆盖掉了外联样式。

<**div class="ui equal width grid" style="width**: 60%; **margin**: 5**px** 5**px** 5**px** 5**px**;**"**>

margin，每一个标签在html是都遵循box model，即盒模型。margin为外边距，代表元素离浏览器上下左右四个边框的距离。padding为内边距，即元素的外边框和它里面内容之间的距离

刷新查看效果。

再来调整颜色，把第一个行中两个列的颜色分别设置为red和blue

<**div class="ui red segment"**>

<**div class="ui blue segment"**>

刷新查看效果

第一行中两个列中放置的是统计的数据，使用semantic ui中的statistic

<http://www.semantic-ui.cn/views/statistic.html>

第二行中的一个列中放置的是主体的元素，使用第三节中用过的divided items

<http://www.semantic-ui.cn/views/item.html#divided>

<**div class="ui equal width grid" style="width**: 60%; **margin**: 5**px** 5**px** 5**px** 5**px**;**"**>  
 <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui red segment"**>  
 <**div class="ui statistic"**>  
 <**div class="value"**>  
 50,000  
 </**div**>  
 <**div class="label"**>  
 Documents  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="column"**>  
 <**div class="ui blue segment"**>  
 <**div class="ui statistic"**>  
 <**div class="value"**>  
 2,000  
 </**div**>  
 <**div class="label"**>  
 Views  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui segment"**>  
 <**div class="ui divided items"**>  
 <**div class="item"**>  
 <**div class="content"**>  
 <**div class="header"**>  
 Title Here!  
 </**div**>  
 <**div class="description"**>  
 Description Here!  
 </**div**>  
 <**div class="extra"**>  
 <**div class="ui label"**>  
 TAG1  
 </**div**>  
 <**div class="ui label"**>  
 TAG2  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>

页面的第一部分就完成了，最后会把这个页面导入到django中，用django的模板进行替换。

来写页面的第二部分。实现一个带有下拉刷新功能的图表。

建一个与上面的ui equal width grid结构平行的分割线结构

再建一个与上面类似的行和列结构

<**div class="ui divider"**></**div**>  
<**div class="ui equal width grid" style="margin**: 5**px** 5**px** 5**px** 5**px**; **width**: 70%;**"**>  
 <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui container segment"**>  
 <**div class="ui container"**></**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>

刷新查看网页，因为下面的列表中没有添加内容，它只显示出一个结构来。下面就来加载highchart的官网。

<http://www.highcharts.com/demo/line-basic>

任意找一个图表，把它复制到我们的网页结构中。

第三周课程中我们已经在jupyter notebook上使用过highchart的图表，只是使用了一个寄托在ipython上的插件来实现的。

在页面底部增加一个script标签，把highchart官网的图表复制过来。

Highcharts 在 4.2.0 开始已经不依赖 jQuery 了，直接用其构造函数既可创建图表

在highchart官网的图表中点“EDIT IN JSFIDDLE”，在编辑器中查看它引用的js代码，把js下载下来，放在我们自己的网页的js文件夹中。

在网页的header引用js文件。

刷新网页，没有出现，因为在网页中没有引入结构。要像官网的EDIT IN JSFIDDLE中看到的那样，在网页中建一个结构。

<div id="container" style="min-width: 310px; height: 400px; margin: 0 auto"></div>

因为highcharts中定义了它的id为“container”

**Highcharts**.**chart**(**'container'**, {

所以必须要在网页中写上如下内容才能实现图表的加载，因为下面div标签中没有写任何内容，浏览这个网页时，图表就在这个标签中被加载出来。

<**div class="ui container" id="container"**></**div**>

下面就去通过下拉列表实现不同图表的切换

<http://www.semantic-ui.cn/modules/dropdown.html>

找到下拉列表的代码，复制到网页中。

<**div class="ui container segment"**>  
 <**div class="ui compact menu"**>  
 <**div class="ui simple dropdown item"**>  
 Dropdown  
 <**i class="dropdown icon"**></**i**>  
 <**div class="menu"**>  
 <**div class="item"**>Choice 1</**div**>  
 <**div class="item"**>Choice 2</**div**>  
 <**div class="item"**>Choice 3</**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="ui container" id="container"**></**div**>  
</**div**>

把名称改变一下，Dropdown: Area, choice: 朝阳，海淀，通州

刷新网页，下拉列表被加载出来了，由于还没有定义加载动作，所以现在不起任何作用。

仿照Menu的加载方式，点击一下Menu，就把sidebar给加载出来了。

<**script**>  
 **$**(**'#menu'**).click(**function** () {  
 **$**(**'.ui.sidebar'**).sidebar(**'toggle'**);  
 });  
</**script**>

同样，定义下拉列表中的任意一个item，就会加载对应的图表。在三个下拉列表的item中定义三个唯一的id

<**i class="dropdown icon"**></**i**>  
 <**div class="menu"**>  
 <**div class="item" id="CY"**>朝阳</**div**>  
 <**div class="item" id="HD"**>海淀</**div**>  
 <**div class="item" id="TZ"**>通州</**div**>  
 </**div**>  
</**div**>

在网站中复制三个不同的图表，放在下面三个script中，并定义三个点击的函数。

<**script**>  
 **$**(**'#CY'**).click(**function** () {  
 **Highcharts**.**chart**(**'container'**, {...});

})

</**script**>

<**script**>  
 **$**(**'#HD'**).click(**function** () {  
 **Highcharts**.**chart**(**'container'**, {...});

})

</**script**>

<**script**>  
 **$**(**'#TZ'**).click(**function** () {  
 **Highcharts**.**chart**(**'container'**, {...});

})

</**script**>

刷新网页，点击下拉列表中的不同item，图表就在下面定义的结构中加载了出来。

<**div class="ui container" id="container"**></**div**>

前提是要传入不同的数据，才会加载出不同的图表和曲线。

接下来要做的事情就是把数据列表部分和图表部分分别汇入模板中。因为上一部分的数据列表部分的数据非常多，图表就会被放在最下面了，这时候把横线分开的两部分分别做两个页面，就要用到模板继承的内容。

本节回顾

这节课我们学习了在Django页面上制作页面和图表的方法。

使用Grid进行了网页的风格布局。

还学习了使用jquery实现网页图表。

我知道你很讨厌写重复的代码，听说程序员都很懒，我们怎么偷懒呢？

使用模板继承。

## 第五节课程 模板继承

第四节中做的网页中由横线分割的部分可以看成是两个独立的网页。这两个网页虽然内容上不同，但是还是有相同的部分。侧栏和顶栏的菜单部分是相同的，称为共用部分，在Django中称为是主模板。通过继承主模板上的部件，这里是侧栏和顶栏的菜单，就可以最大程度的减少代码的重复量。

主模板就相当于在一块木板上挖出一些不同形状的坑。子模板就相当于是照着坑的形状设计单个的页面，这些页面上的特征是独一无二的，是主模板上所没有的。这时按着主模板上预留好的位置，访问子模板的时候，子模板会根据继承关系把主模板拉过来，找到子模板对应的位置，加载出整个网页。这样就最大程度上复用了主模板上的代码，以上就是模板继承的基本原理。

第一步 static修改

把做好的网页导入到Django中，改名为new\_base.html，并与static进行关联

同第三节课程中类似，把相应的文件复制到static文件夹中，网页放入templates中。

并把所有有引用的部分用static来替换。包括css, js, 图片。

第二步 在主模板上挖洞

要做两个网页，一个是用来展示图表的，一个是用来加载分页的基础信息。先分析一下主模板中哪些部分是固定不变的。

head部分中加载的js，css等内容的部分是必要的不变的。

body中的sidebar和Menu都是要留下的。

唯一要变的就是pusher中的ui equal width grid部分，即定义的等宽的栏的部分。把其中的内容剪切到一个名为index\_data.html的新的模板中。

删除<**div class="ui divider"**></**div**>

把下面关于图表的ui equal width grid再次剪切到一个名为chart.html的模板中。

第三步 进行主模板和子模板的关联

第二步是在主模板上挖洞，挖完洞后要把它先填上。回到主模板中，把空出来的部分用模板的标签把它填上。

{% **block content** %}  
   
{% **endblock** %}

在以后访问index\_data网页时，会把该填充的部分从主模板上找到并填充上。

static标签则是Django系统自带的，它就是用来引用静态目录的。而此处的content标签是自定义的，叫什么名字都可以。也可以改为

{% **block grid** %}  
   
{% **endblock** %}

现在位置已经占好了。接下来的任务回到子模板中，子模板是以后要访问的内容，还要把它和主模板进行一下关联。要在第一行中说明缺失的一部分去哪里找。写为如下的形式：

{% **extends 'new\_base.html'** %}  
{% **block grid** %}  
<**div class="ui equal width grid" style="margin**: 5**px** 5**px** 5**px** 5**px**; **width**: 70%;**"...**>  
{% **endblock** %}

还需要进行修改的地方是50,000和2,000这两个信息。应该是引用数据库并从中获得的统计信息。还要把title, description和tag的内容从数据库中引用信息来进行替代，与第二节中的操作方法类似。

<**div class="content"**>  
 <**div class="header"**>  
 Title Here!  
 </**div**>  
 <**div class="description"**>  
 Description Here!  
 </**div**>  
 <**div class="extra"**>  
 <**div class="ui label"**>  
 TAG1  
 </**div**>  
 <**div class="ui label"**>  
 TAG2  
 </**div**>  
 </**div**>  
</**div**>

还要在页面最后面加上一个分页符

切换回models层，进行一下数据层面的修改。

使用我们真实数据库中的数据。

**from** django.db **import** models  
**from** mongoengine **import** \*  
  
**class** ItemInfo(Document):  
 title = StringField()  
 url = StringField()  
 pub\_data = StringField()  
 area = ListField(StringField())  
 cates = ListField(StringField())  
 look = StringField()  
 time = StringField()  
 price = IntField()  
 meta = {**'collection'**: **'item\_infoS'**}

接着在view中调整我们的视图，把其中所有的ArtiInfo改为ItemInfo。

context中多了两个参数。counts和last\_time。counts是数据库中数据的总数。last\_time是把筛选出来的pub\_date用mongoengine的order\_by的方法进行一个分类。-pub\_date表示从近往远排列的方式。limit(1)就是只取一个，结合-pub\_date的排序就是只取最新的日期。

render函数中需要request, template和context，把template的参数修改为index\_data.html，也就是说访问我们的子页面时会找到主模板进行加载。

**from** django.shortcuts **import** render  
**from** django\_web.models **import** ItemInfo  
**from** django.core.paginator **import** Paginator  
  
**def** index(request):  
 limit = 5  
 arti\_info = ItemInfo.objects  
 paginator = Paginator(arti\_info, limit)  
 page = request.GET.get(**'page'**,1)  
loaded = paginator.page(page)  
 context = {  
 **'ItemInfo'**: loaded,  
 **'counts'**: arti\_info.count(),  
 **'last\_time'**: arti\_info.order\_by(**'-pub\_date'**).limit(1),  
 }  
 **return** render(request, **'index\_data.html'**, context)

在template的index\_data.html中使用循环的方法进行重构，加载数据库中的真实的数据。

把50000的数据改为{{ counts }}，主体部分改为如下内容

<**div class="ui container segment"**>  
 <**div class="ui divided items"**>  
 {% **for item in ItemInfo** %}  
 <**div class="item"**>  
 <**div class="content"**>  
 <**a class="header"**>  
 {{ **item**.**title** }}  
 </**a**>  
 <**div class="description"**>  
 {{ **item**.**area** }}  
 </**div**>  
 <**div class="extra"**>  
 {% **for tag in item**.**cates** %}  
 <**div class="ui label"**>  
 {{ **tag** }}  
 </**div**>  
 {% **endfor** %}  
 </**div**>  
 </**div**>  
 </**div**>  
 {% **endfor** %}  
 </**div**>  
</**div**>

把数据库中装载的信息全部放在ItemInfo中，用context传回来，用一个循环把ItemInfo中的内容取出来，并分别赋值给相应的位置。

然后把分页补上。还是如之前课程中的，使用一个简单的逻辑控制语句，如果有前面的页，就可以点击跳转到前面的页的按钮并配上上一页的链接，如果有后面的页，就可以点击跳转到后面的页的按钮并配上下一页的链接。中间显示当前的页数。

把setting中的连接数据库改为ceshi

python3 manage.py runserver

刷新网页查看。

可以把地区也用循环的方式取出来，变成如类目一样的标签。

如上便完成了第一个页面。接下来要做图表的页面。

先写继承，再修改内容。

因为子模板中放的独一无二的内容，不是共用的部分。在new\_base中保留控制sidebar行为的script，因为它在所有的子模板中是公用的，而把new\_base.html中script部分的数据给剪切到chart.html中。剪切的时候还是相当于挖坑，挖完了坑要写上挖走了什么，把原来highcharts的js代码位置上写上如下内容。

{% **block chartjs** %}  
{% **endblock** %}

而粘贴到chart.html中时也不能按原本html的方法进行粘贴，而是用“填坑”的方法。建一个block chartjs的内容。

{% **block chartjs** %}  
 <**script...**>  
 <**script...**>  
 <**script...**>  
{% **endblock** %}

从总体上来看，chart.html模板中就只有两个块的内容。block grid, block chartjs。在访问这个页面时，会按照这两个块的名称去主模板中加载相应的内容。

{% **block grid** %}  
<**div class="ui equal width grid" style="..."**>  
{% **endblock** %}

{% **block chartjs** %}  
 <**script...**>  
 <**script...**>  
 <**script...**>  
{% **endblock** %}

由于与index\_data是不同的页面，需要给它分配一个视图函数，在views.py中要加上视图函数。

**def** chart(request):  
 **return** render(request, **'chart.html'**)

直接使用render函数就可以了。render函数中返回request和新的chart.html页面。context的内容暂时不写，之后还要再做修改，在chart中渲染出数据库中真实的数据来。而在之间复制到chart中的内容都可以看成是暂时占用的数据。

最后还需要给chart来创建一个链接。

在urls.py中创建一个新的url

**from** django.conf.urls **import** url  
**from** django.contrib **import** admin  
**from** django\_web.views **import** index,chart  
  
urlpatterns = [  
 url(**r'^admin/'**, admin.site.urls),  
 url(**r'^index/'**, index),  
 url(**r'^chart'**, chart),  
]

打开浏览器查看结果。

index\_data和chart两个模板就相当于两个不同的网页。

还应该在sidebar的Document和charts中创建一个链接到这两个网页上的链接。

下面就把真实的数据库中的数据填到网页中。

可以使用第三周大作业中创建图表时使用的pipeline1，但使用方法稍有不同。

pipeline1 = [

{'$match':{'$and':[{'pub\_date':{'$gte':'2015.12.25','$lte':'2015.12.27'}},{'area':{'$all':['朝阳']}}]}},

{'$group':{'\_id':{'$slice':['$cates',2,1]},'counts':{'$sum':1}}},

{'$limit':3}

]

**for** i **in** item\_info.aggregate(pipeline1):

print(i)

根据上面的方式，来调整views视图中chart部分的context。

以chart(request)这个函数中是需要传入context参数的。先在上面定义视图函数。

#不同地区发帖量前三名

**def** topx(date1,date2,area,limit):  
  
 pipeline = [  
 {**'$match'**:{**'$and'**:[{**'pub\_date'**:{**'$gte'**:date1,**'$lte'**:date2}},{**'area'**:{**'$all'**:area}}]}},  
 {**'$group'**:{**'\_id'**:{**'$slice'**:[**'$cates'**,2,1]},**'counts'**:{**'$sum'**:1}}},  
 {**'$limit'**:limit},  
 {**'$sort'**:{**'counts'**:-1}}  
 ]  
  
 **for** i **in** ItemInfo.\_get\_collection().aggregate(pipeline):  
 data = {  
 **'name'**: i[**'\_id'**][0],  
 **'data'**: [i[**'counts'**]],  
 **'type'**: **'column'** }  
 **yield** data

series = [i **for** i **in** topx(**'2015.12.25'**,**'2015.12.27'**,[**'朝阳'**],3)]

这时用yield data返回的是一个生成器，为了加快加载速度，也可以直接返回生成器生成的结果，这样就不用每次访问页面就去从数据库中调用生成器生成数据了。不过这只对静态的数据库有效。

生成三个不同地区的series数据。

series\_CY = [i **for** i **in** topx(**'2015.12.25'**,**'2015.12.27'**,[**'朝阳'**],3)]  
series\_TZ = [i **for** i **in** topx(**'2015.12.25'**,**'2015.12.27'**,[**'通州'**],3)]  
series\_HD = [i **for** i **in** topx(**'2015.12.25'**,**'2015.12.27'**,[**'海淀'**],3)]

把上面生成的series依次传入到chart的context中。

**def** chart(request):  
 context = {  
 **'chart\_CY'**:series\_CY,  
 **'chart\_TZ'**:series\_TZ,  
 **'chart\_HD'**:series\_HD,  
 }  
 **return** render(request,**'chart.html'**,context)

然后在charts子模板中对script部分进行修改。至于怎么修改，如何定制图表。

<http://api.highcharts.com/highcharts>

图表中所有的选项都可以在这里找到。根据需要选择相应的选项即可。

<https://api.hcharts.cn/>

{% **block chartjs** %}  
 <**script**>  
 **$**(**'#CY'**).click(**function** () {  
 **Highcharts**.**chart**(**'container'**, {  
 **credits**:{  
 **enabled**:**false** },  
 **title**: {  
 **text**: **'朝阳二手交易'** },  
 **yAxis**: {  
 **title**: {  
 **text**: **'TOP 3'** }  
 },  
 **series**: {{ **chart\_CY** }}  
 });  
 });  
 </**script**>  
 <**script**>  
 **$**(**'#HD'**).click(**function** () {  
 **Highcharts**.**chart**(**'container'**, {  
 **credits**:{  
 **enabled**:**false** },  
 **title**: {  
 **text**: **'海淀二手交易'** },  
 **yAxis**: {  
 **title**: {  
 **text**: **'TOP 3'** }  
 },  
 **series**: {{ **chart\_HD** }}  
 });  
 });  
 </**script**>  
 <**script**>  
 **$**(**'#TZ'**).click(**function** () {  
 **Highcharts**.**chart**(**'container'**, {  
 **credits**:{  
 **enabled**:**false** },  
 **title**: {  
 **text**: **'通州二手交易'** },  
 **yAxis**: {  
 **title**: {  
 **text**: **'TOP 3'** }  
 },  
 **series**: {{ **chart\_TZ** }}  
 });  
 });  
 </**script**>  
{% **endblock** %}

打开网页查看，点击下拉列表无法加载图表，右键检查，发现script中所有的series中传进去的序列都是乱码显示的，是utf-8编码的问题。使用模板语言中一个很方便的方法，即过滤器，在series后面的加上safe过滤器，chart\_CY|safe，safe的意思是过滤文本，使其不去转译。

刷新网页，就加载出来了数据库中动态的数据。

课程回顾

这节课我们学习了模板继承

用小时候玩的积木，理解主模板和子模板的关系

子模板把主模板拉过来，套到相应位置里，就相当于模板继承。

学了这么多，马上就要完成最终成果了。

来挑战本周的实战作业吧。

## 第四周实战作业 实现二手行情网站

使用二手行情的网站，实现几个图表。

第一个是各类目的发帖量对比，做一个柱状图。发帖问题柱状图。

第二个是对于一天内交易出去的物品，统计其品类分布和地区分布，看看什么样的商品热销。作两个饼状图。一天内交易物品种类分布饼状图，一天内交易物品地区分布饼状图。

第三个是统计各个城区中，最热销的三个类目是什么。在Area下拉列表中可以切换不同的城区。

需要用到的组件请参考semantic ui官方文档

## 第四周实战作业讲解

第一个是各类目的发帖量对比，做一个柱状图。发帖总量柱状图。

其实在第三周中已经做出来了，只不过现在再用highcharts再实现一遍。

第二个是对于一天内交易出去的物品，统计其品类分布和地区分布，看看什么样的商品热销。作两个饼状图。一天内交易物品种类分布饼状图，一天内交易物品地区分布饼状图。

第三周也做过类似的饼图。

先看模板层中，在chart.html中，设定一个等宽的网格，网格中是一行行的内容，每一行中就存储着图表显示的区域，一行对应于一个图表。

{% **block grid** %}  
<**div class="ui equal width grid" style="margin**:5**px** 0 5**px** 0;**width**:70%**"**>  
*{# ------------------------------------------------ CHART BLOCK --------------------------------------------------- #}* <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui container segment"**>  
 <**div class="ui container" id="chart2"**></**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
*{# ------------------------------------------------ CHART BLOCK --------------------------------------------------- #}* <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui container segment"**>  
 <**div class="ui container" id="pie1"**></**div**>  
 </**div**>  
 </**div**>  
  
 <**div class="column"**>  
 <**div class="ui container segment"**>  
 <**div class="ui container" id="pie2"**></**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
*{# ------------------------------------------------ CHART BLOCK --------------------------------------------------- #}* <**div class="row"**>  
 <**div class="column"**>  
 <**div class="ui container segment"**>  
 <**div class="ui compact menu"**>  
 <**div class="ui simple dropdown item"**>  
 Area  
 <**i class="dropdown icon"**></**i**>  
 <**div class="menu"**>  
 <**div class="item" id="CY"**>朝阳</**div**>  
 <**div class="item" id="HD"**>海淀</**div**>  
 <**div class="item" id="TZ"**>通州</**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="ui container" id="chart1"**></**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
*{# ------------------------------------------------ CHART BLOCK --------------------------------------------------- #}*</**div**>  
{% **endblock** %}

第一个row中的数据为发帖总量柱状图的显示控制。第二个row中的数据又分为两列，每列中都有一个饼图，是一天内交易物品种类分布和一天内交易物品地区分布两个饼图。第三个row中的数据就是第四节中用Django所做的可以用下拉列表控制的图表。

切换到视图层，看视图层中是如何处理的。

第四周的课程作业只是把以前所学的所有的的知识都串了起来。

视图层中只不过是三个生成器，如发帖总量柱状图是把第三周第二节中的内容用pipeline的方法重写了一下。当初是用了一个find函数加上一个列表循环的方法要需要的数据提取了出来，这里直接使用pipeline的方法直接把数据取出来了。

**def** total\_post():  
 pipeline = [  
 {**'$group'**:{**'\_id'**:{**'$slice'**:[**'$cates'**,2,1]},**'counts'**:{**'$sum'**:1}}},  
 ]  
  
 **for** i **in** ItemInfo.\_get\_collection().aggregate(pipeline):  
 print(i)  
 data = {  
 **'name'**:i[**'\_id'**][0],  
 **'y'**:i[**'counts'**]  
 }  
 **yield** data  
  
series\_post = [i **for** i **in** total\_post()]

pipeline是直接进行分组操作，组中元素是对cates种类进行分片得到的。从cates中第二个元素开始切出来一个元素。还把它的统计项分成了一个组。每找到一个这样的元素，counts就加1。

然后就是使用mongoengine中的聚合管道按照一定的格式把所有的data给取出来。

最后用一个列表的解析式把取出来的结构化的数据保存在一个变量中。

**def** one\_day\_deal\_cate():  
 pipeline = [  
 {**'$match'**:{**'$and'**:[{**'pub\_date'**:{**'$gte'**:**'2015.12.25'**,**'$lte'**:**'2016.01.11'**}},{**'time'**:1}]}},  
 {**'$group'**:{**'\_id'**:{**'$slice'**:[**'$cates'**,2,1]},**'counts'**:{**'$sum'**:1}}},  
 {**'$sort'**:{**'counts'**:1}}  
 ]  
  
 **for** i **in** ItemInfo.\_get\_collection().aggregate(pipeline):  
 data = {  
 **'name'**:i[**'\_id'**][0],  
 **'y'**:i[**'counts'**]  
 }  
 **yield** data

在一天内交易的类目分布图中。

先在pipeline中过滤出在一定时间段内交易时长为1天的数据。然后进行分组操作，再按照降序排列。因为要做的是饼图，用降序排列可以使图表更加美观，故这里用降序的排列方式。

而视图函数chart(request)中把其它的图表信息加上即可。

**def** chart(request):  
 context = {  
 **'chart\_CY'**:series\_CY,  
 **'chart\_TZ'**:series\_TZ,  
 **'chart\_HD'**:series\_HD,  
 **'series\_post'**:series\_post,  
 **'pie1\_data'**:pie1\_data,  
 **'pie2\_data'**:pie2\_data  
 }  
 **return** render(request,**'chart.html'**,context)

切换回模板，查看js部分。

highcharts的使用方法，在<https://api.hcharts.cn/>网站上查看官方文档，理解作出来的图表中每一个元素和控制参数是什么，然后在官网上找一个实例，把它修改为符合我们要求的代码即可，删除不需要的部分，把一些标签和数据修改为我们自己的即可。

课程回顾

这节课我们完成了大作业，搭建了一个数据可视化的Django网站。

由于知识在前面讲过了，所以可以轻松完成。

在这四周里，我们不仅掌握了爬虫、MongoDB、Django……

更重要的是，我们掌握了编程这种思维方式。从此以后，可以运用编程做更多的项目，学以致用。

## 第四周补充资料

Djangogirl学习指南

<https://djangogirlstaipei.gitbooks.io/django-girls-taipei-tutorial/content/index.html>

跟着课程做一遍，能完成一个旅游日记的网站。

DjangoBook

<http://djangobook.py3k.cn/2.0/>

<http://djangobook.com/>

解读：Djangobook是公认的比较全面详尽的Django文档，你的大多数问题都能在里面得到解答，但弊端在于，对于初学者比较难懂。不过在课程之后再学应该会好些。有决心学好Django的话值得啃一遍。

Semantic UI官方文档

<http://www.semantic-ui.cn>

视频中已经提到过很多次了，需要实现哪个功能，去官方文档里面对照着示例实现就可以了。

## 推荐10本编程书

为你推荐10本有趣的编程书，让你在课程之后继续深入学习，探索编程的乐趣。

**1编码 隐匿在计算机软硬件背后的语言**

和我们课程的风格差不多，用了很多生活中的例子由浅入深的讲解专业知识，包括：二进制，操作系统、计算机组成原理等等。

**2图解http**

有的书让你看了有‘一览众山小‘的感觉，有时山太高，很多人就不爬。有的书是给了你一个上山的梯子，比如这本

**3大话数据结构**

第一本数据结构方面的书，作者用了大量的实例，图和吐槽来直观解释这些东西是什么，能干什么。

**4欧姆社学习漫画 漫画数据库**

当叫数据库的小精灵告别主角时，我也很留恋，真希望能一直学下去

**5鸟哥的Linux私房菜**

书中主要讲了Linux的各种操作，讲解得很详细，是从鸟哥学习的角度来讲的。可以放在案头当工具书来用。

**6程序员的思维修炼**

这些编程思维的方法论，不仅仅是程序员，所有想要了解大脑认知过程并且发掘潜能的人都可以读读。

**7编程大师访谈录**

虽说高手不可模仿，但我们无疑可以受到不小的启发。如果能被他们打动，分享同样的激情，也算一庄美事。

**8黑客与画家**

真正意义上的黑客精神，创业，编程语言，是这本技术散文集的三个主题。

**9利用python进行数据分析**

本书作者也是pandas库的作者，这是一本介绍如何用python控制、处理、整理、分析结构化数据，并有大量的实践安全的书。唯一的缺点大概是比较难民吧。

**10程序是怎样跑起来的**

这本书讲解了程序的运行原理，用图文并茂的方式力争让小学生、老奶奶也能读懂。