高德API+Python解决租房问题

**实验环境：**

用户名：lvze

密码：None

**高德API+Python解决租房问题**

**一、课程介绍**

**1. 课程背景**

课程来自一段租房血泪史：说白了就是都没有钱，于是打算合租。因为穷所以不可能找有门店的的中介，只能看看赶集、58、和一些租房APP。期间需要考虑两个人的通勤范围来选地段，由于对交通的不熟悉，只有选择自己附近的较贵的地段，花了很多时间阅览赶集或者58里的个人房源信息，然而个人房源信息中仍充斥着大量中介，抱着一点希望打了几个电话，得到的回答都是这个价位根本租不到，再高点也租不到（大都与发布的房源信息不符）。最后终于还是在宿舍关闭前一个星期租到一个性价比还可以的隔断。毕竟隔断还是不方便的，所以打算在室友找到工作后换一个新地方，于是就有了这个租房脚本和课程。

相信也有不少的应届毕业生可能会遭遇同样的境况，希望这门课能真的帮到大家，也许不光是在租房子方面。推荐阅读知乎上的一个问题：[你有哪些用计算机技能解决生活问题的经历？](https://www.zhihu.com/question/28361295" \t "_blank)

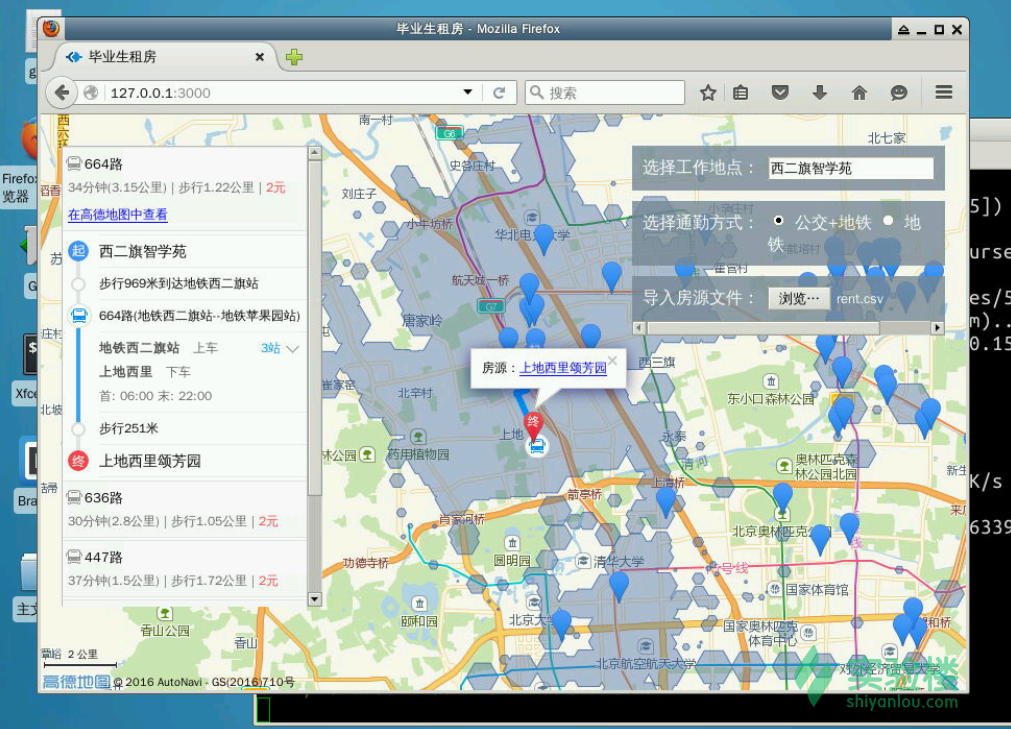
总结一下租房难的症结：

* 没钱。
* 小中介发布的价位一般都是假的，会浪费你很多时间。
* 对交通路线不熟悉以致于选择面窄。
* 如果是多人，得同时考虑多人的通勤时间。

本课程将解决的问题：

* 学习了技术，增长了知识，就能找到好工作，找到好工作就能有钱。
* 这次选的房源信息来自58的品牌公寓馆，所以没有那种小中介，价位就我和我室友来说可以接受。其实可以做个分类器过滤赶集上的中介来找低价个人房源的，有需要的同学可以试一下。
* 通勤范围在地图上圈出，解决了对交通路线不熟悉的问题
* 本课程是单人版的，但代码中只要删掉一个语句就能当多人用了（但是路径规划的功能是只能给一个人用）。如果是直接拿来多人用的话，还是开多个页面比较好。

最终效果图如下：



由于没做前端兼容，导致右上角崩了。自用的小工具其实也不用整多好看，效率第一。

如图，划出来的大片蓝色色块就是距离工作地点一小时车程内的区域。蓝色的标记就是房源，点击后会自动给出路径规划和房源地址。红色标记（不是"终"）是工作地点,在图里被挡住了。工作地点的输入框有自动补完的功能，也是很方便的。至于房源文件我们会通过编写Python脚本在抓取房源信息后生成。

**2. 课程知识点**

本课程项目完成过程中，我们将学习：

1. requests、BeautifulSoup、csv 等库的简单使用
2. 高德地图 Javascript API 的使用

**二、实验环境**

打开终端，进入 Code 目录，创建 rent\_proj 文件夹, 并将其作为我们的工作目录。

$ cd Code

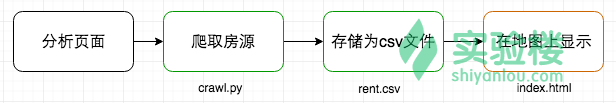
$ mkdir rent\_proj && cd rent\_proj

安装需要的库：

$ sudo apt-get install python-bs4

**三、实验原理**

实验中会用到三个文件：crawl.py，rent.csv与index.html，其中rent.csv由crawl.py生成，是房源文件。crawl.py是一个非常简单的爬取网页的脚本。index.html是最重要的显示地图的部分。实现的流程大致如下：



我为什么不把js代码和css代码从index.html中分出来呢，写脚本怎么顺手怎么来就好。

代码没有难度，主要就是看看几个API如何使用，下面给出文档链接：

* [高德 JavaScript API 帮助文档](http://lbs.amap.com/api/javascript-api/summary/)
* <http://lbs.amap.com/api/javascript-api/summary/>
* [高德 JavaScript API 示例中心](http://lbs.amap.com/api/javascript-api/example/map/map-show/)
* <http://lbs.amap.com/api/javascript-api/example/map/map-show/>
* [Requests: HTTP for Humans](http://www.python-requests.org/en/master/)
* <http://www.python-requests.org/en/master/>
* Beautiful Soup 4.2.0 文档<https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/>
* [csv — CSV 文件的读写](http://python.usyiyi.cn/python_278/library/csv.html) <http://python.usyiyi.cn/python_278/library/csv.html>

**四、实验步骤**

**1. 分析页面**

先分析一下我们需要爬取的页面：<http://bj.58.com/pinpaigongyu/>

选择好目标价位：



打开查看器：

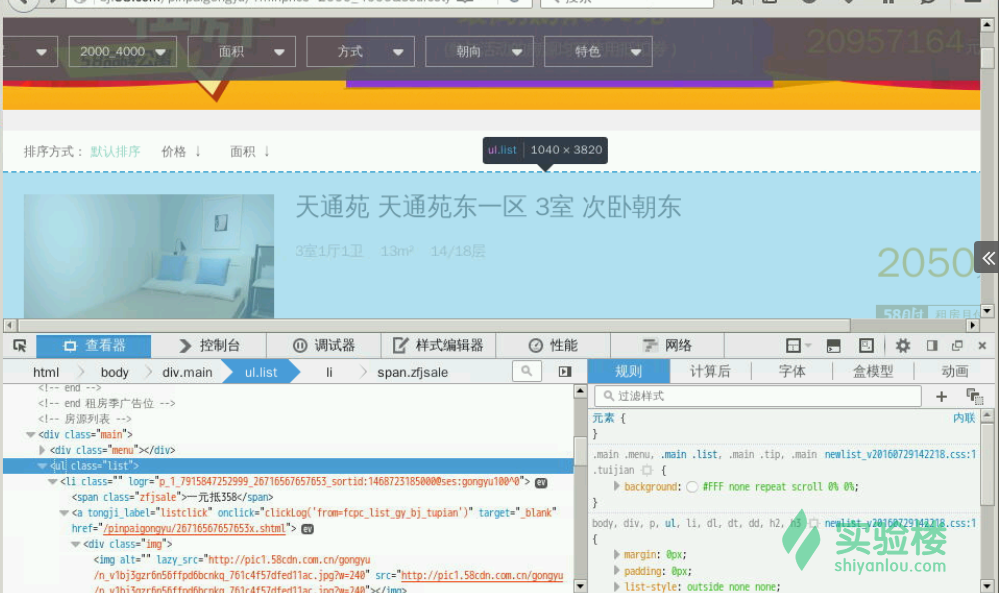


审查元素，分页上的 1 2 3 4 5 随便选一个：



大致了解了它的路径规则：/pingpaigongyu/pn/{page}/minprice={min\_rent}\_{max\_rent}

对比第一页和第一千页：



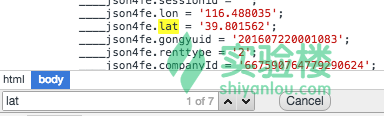


经过观察，决定将页面有无.list元素来作为是否已读取完所有房源的判断条件。

看一下页面能够提供给我们什么信息：



框框圈出的信息对我们来说已经足够了。当然房源的具体经纬度也可以点进去查看代码得到：



简便起见每一个房源就只记录以下信息：[房源名称，地址，月租，房源url地址]。

其中地址直接取房源名称里的小区名，有的房源第二列是公寓名而不是小区名，那就只能取第一列作为地址了，有些公寓如果合并第一列第二列地图上会搜不到。

**2. 编写Python脚本**

在工作目录下创建crawl.py文件，这里先直接给出全部代码。

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

from bs4 import BeautifulSoup

from urlparse import urljoin

import requests

import csv

url = "http://bj.58.com/pinpaigongyu/pn/{page}/?minprice=2000\_4000"

#已完成的页数序号，初时为0

page = 0

csv\_file = open("rent.csv","wb")

csv\_writer = csv.writer(csv\_file, delimiter=',')

while True:

page += 1

print "fetch: ", url.format(page=page)

response = requests.get(url.format(page=page))

html = BeautifulSoup(response.text)

house\_list = html.select(".list > li")

# 循环在读不到新的房源时结束

if not house\_list:

break

for house in house\_list:

house\_title = house.select("h2")[0].string.encode("utf8")

house\_url = urljoin(url, house.select("a")[0]["href"])

house\_info\_list = house\_title.split()

# 如果第二列是公寓名则取第一列作为地址

if "公寓" in house\_info\_list[1] or "青年社区" in house\_info\_list[1]:

house\_location = house\_info\_list[0]

else:

house\_location = house\_info\_list[1]

house\_money = house.select(".money")[0].select("b")[0].string.encode("utf8")

csv\_writer.writerow([house\_title, house\_location, house\_money, house\_url])

csv\_file.close()

鉴于爬的量不大所以就简单处理了。

csv一般用作表格文件，直接用文本编辑器打开也可读，行与行之间就是换行来隔开，列与列之间就是用逗号（也可指定其它字符）隔开，Python标准库中的csv库就是用来读写csv文件的。

这里只需要写csv文件：

#导入csv

import csv

# 打开rent.csv文件

csv\_file = open("rent.csv","wb")

# 创建writer对象，指定文件与分隔符

csv\_writer = csv.writer(csv\_file, delimiter=',')

# 写一行数据

csv\_writer.writerow([house\_title, house\_location, house\_money, house\_url])

#关闭文件

csv\_file.close()

requests是一个对使用者非常友好的http库，看一遍[Quickstart](http://www.python-requests.org/en/master/user/quickstart/" \t "_blank)就能基本掌握它的使用。

用到它的地方也就仅仅两句：

# 抓取目标页面

response = requests.get(url.format(page=page))

# 获取页面正文

response.text

Beautiful Soup是一个用来解析html或者xml文件的库，支持元素选择器，使用起来也非常方便：

# 创建一个BeautifulSoup对象

html = BeautifulSoup(response.text)

# 获取class=list的元素下的所有li元素

house\_list = html.select(".list > li")

# 得到标签包裹着的文本

house.select("h2")[0].string.encode("utf8")

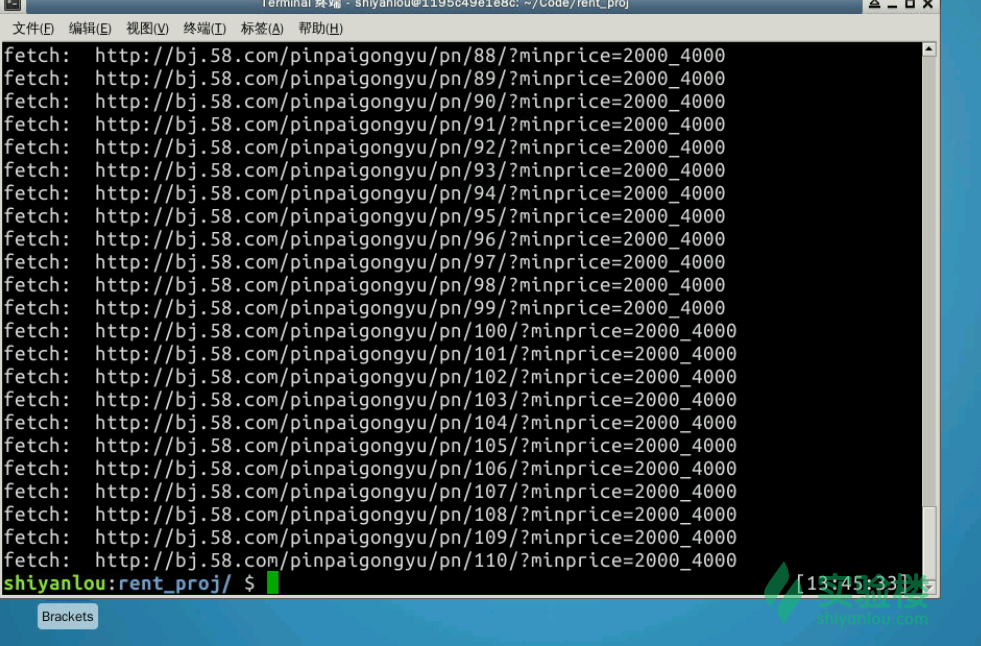
# 得到标签内属性的值

house.select("a")[0]["href"]

由于读取到的链接路径是相对路径，所以需要urljoin得到完整的url地址。

urljoin(url, house.select("a")[0]["href"])

讲解完毕，运行代码python crawler.py：



可能要1分半钟，可以先去玩一局扫雷。

笔者用自己的电脑跑差不多是20多秒。

此时目录下已生成了rent.csv

注意如果你太频繁地抓取页面IP可能会被屏蔽，那样就抓不到页面而是直接报错了。如果抓取不成功的话，可以使用这个文件继续接下来的实验：

$ wget https://labfile.oss.aliyuncs.com/courses/599/rent.csv

**3. 设计页面**

页面大框架可直接从示例中心复制：[高德 JavaScript API 示例中心](http://lbs.amap.com/api/javascript-api/example/map/map-show/)

这里先给出全部的设计代码，新建index.html复制粘贴下面的代码：

<html>

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="initial-scale=1.0, user-scalable=no, width=device-width">

<title>毕业生租房</title>

<link rel="stylesheet" href="http://cache.amap.com/lbs/static/main1119.css" />

<link rel="stylesheet" href="http://cache.amap.com/lbs/static/jquery.range.css" />

<script src="http://cache.amap.com/lbs/static/jquery-1.9.1.js"></script>

<script src="http://cache.amap.com/lbs/static/es5.min.js"></script>

<script src="http://webapi.amap.com/maps?v=1.3&key=22d3816e107f199992666d6412fa0691&plugin=AMap.ArrivalRange,AMap.Scale,AMap.Geocoder,AMap.Transfer,AMap.Autocomplete"></script>

<script src="http://cache.amap.com/lbs/static/jquery.range.js"></script>

<style>

.control-panel {

position: absolute;

top: 30px;

right: 20px;

}

.control-entry {

width: 280px;

background-color: rgba(119, 136, 153, 0.8);

font-family: fantasy, sans-serif;

text-align: left;

color: white;

overflow: auto;

padding: 10px;

margin-bottom: 10px;

}

.control-input {

margin-left: 120px;

}

.control-input input[type="text"] {

width: 160px;

}

.control-panel label {

float: left;

width: 120px;

}

#transfer-panel {

position: absolute;

background-color: white;

max-height: 80%;

overflow-y: auto;

top: 30px;

left: 20px;

width: 250px;

}

</style>

</head>

<body>

<div id="container"></div>

<div class="control-panel">

<div class="control-entry">

<label>选择工作地点：</label>

<div class="control-input">

<input id="work-location" type="text">

</div>

</div>

<div class="control-entry">

<label>选择通勤方式：</label>

<div class="control-input">

<input type="radio" name="vehicle" value="SUBWAY,BUS" onClick="takeBus(this)" checked/> 公交+地铁

<input type="radio" name="vehicle" value="SUBWAY" onClick="takeSubway(this)" /> 地铁

</div>

</div>

<div class="control-entry">

<label>导入房源文件：</label>

<div class="control-input">

<input type="file" name="file" onChange="importRentInfo(this)" />

</div>

</div>

</div>

<div id="transfer-panel"></div>

<script>

var map = new AMap.Map("container", {

resizeEnable: true,

zoomEnable: true,

center: [116.397428, 39.90923],

zoom: 11

});

</script>

</body>

</html>

讲解一下部分代码：

这一句中你会看到key这个参数，它需要你注册高德的开发者用户，创建应用才能得到，由于 JS API 不像它家的 Web API 一样有流量限制，所以这个key大可随意使用。

<script src="http://webapi.amap.com/maps?v=1.3&key=22d3816e107f199992666d6412fa0691&plugin=AMap.ArrivalRange,AMap.Scale,AMap.Geocoder,AMap.Transfer,AMap.Autocomplete"></script>

载入编写代码时可能用到的API插件：

<script src="http://webapi.amap.com/maps?v=1.3&key=22d3816e107f199992666d6412fa0691&plugin=AMap.ArrivalRange,AMap.Scale,AMap.Geocoder,AMap.Transfer,AMap.Autocomplete"></script>

从名字就可以看出插件的作用：

* ArrivalRange：公交到达圈
* Scale：标尺
* Geocoder：正向地理编码（地址-坐标）
* Transfer：路径规划
* Autocomplete：地址自动补全

这些都可以在示例中心找到对应的例子

在输入标签内可以设定onClick与onChange属性，它们的作用是当该输入元素上发生点击或者内容变化的事件时，设定的内容就会被运行。

<input type="radio" name="vehicle" value="SUBWAY,BUS" onClick="takeBus(this)" checked/>

<input type="file" name="file" onChange="importRentInfo(this)" />

control-panel 就是右上角的输入面板区域。transfer-panel是路径规划面板，它只有在调用了路径规划的函数时才会出现。

<div class="control-panel">

<div id="transfer-panel">

在页面设计的部分保留了部分js代码是为了页面好看点。

var map = new AMap.Map("container", {

resizeEnable: true, //页面可调整大小

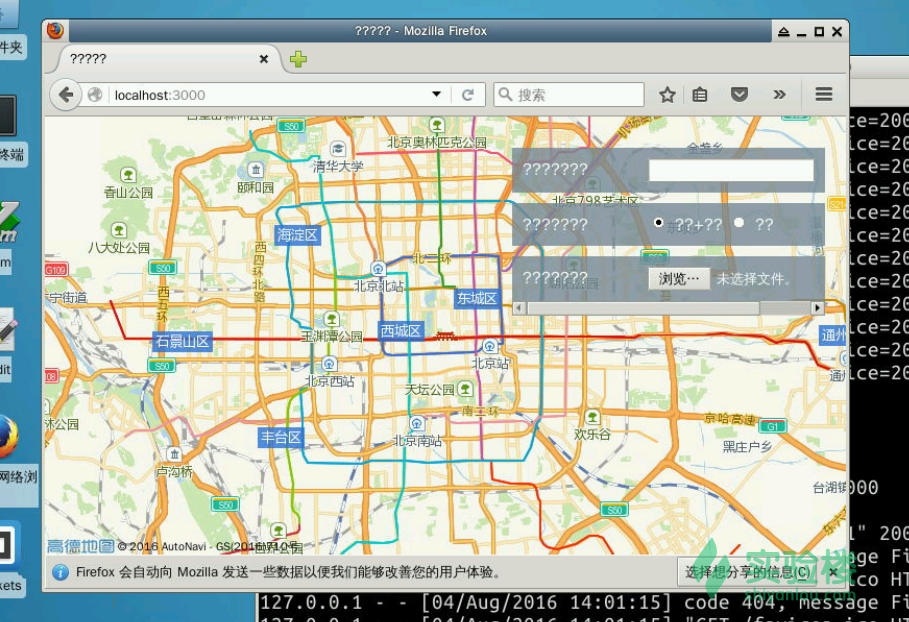
zoomEnable: true, //可缩放

center: [116.397428, 39.90923], //地图中心，这里使用的是北京的经纬度

zoom: 11 //缩放等级，数字越大离地球越近

});

输入python -m SimpleHTTPServer 3000打开服务器，浏览器访问localhost:3000查看效果：



由于无法复制中文的缘故，所以界面上是???，这之后会放出完整的文件链接。

**4. 调用高德 API**

高德这套API只要逛逛示例中心，示例中心没有的就去看看参考手册，基本就够用了。

同样先放上全部代码，之后再分解着讲，分解的部分会尽量给出高德的实例作参考。

接着上一节index.html里script的内容：

var map = new AMap.Map("container", {

resizeEnable: true,

zoomEnable: true,

center: [116.397428, 39.90923],

zoom: 11

});

var scale = new AMap.Scale();

map.addControl(scale);

var arrivalRange = new AMap.ArrivalRange();

var x, y, t, vehicle = "SUBWAY,BUS";

var workAddress, workMarker;

var rentMarkerArray = [];

var polygonArray = [];

var amapTransfer;

var infoWindow = new AMap.InfoWindow({

offset: new AMap.Pixel(0, -30)

});

var auto = new AMap.Autocomplete({

input: "work-location"

});

AMap.event.addListener(auto, "select", workLocationSelected);

function takeBus(radio) {

vehicle = radio.value;

loadWorkLocation()

}

function takeSubway(radio) {

vehicle = radio.value;

loadWorkLocation()

}

function importRentInfo(fileInfo) {

var file = fileInfo.files[0].name;

loadRentLocationByFile(file);

}

function workLocationSelected(e) {

workAddress = e.poi.name;

loadWorkLocation();

}

function loadWorkMarker(x, y, locationName) {

workMarker = new AMap.Marker({

map: map,

title: locationName,

icon: 'http://webapi.amap.com/theme/v1.3/markers/n/mark\_r.png',

position: [x, y]

});

}

function loadWorkRange(x, y, t, color, v) {

arrivalRange.search([x, y], t, function(status, result) {

if (result.bounds) {

for (var i = 0; i < result.bounds.length; i++) {

var polygon = new AMap.Polygon({

map: map,

fillColor: color,

fillOpacity: "0.4",

strokeColor: color,

strokeOpacity: "0.8",

strokeWeight: 1

});

polygon.setPath(result.bounds[i]);

polygonArray.push(polygon);

}

}

}, {

policy: v

});

}

function addMarkerByAddress(address) {

var geocoder = new AMap.Geocoder({

city: "北京",

radius: 1000

});

geocoder.getLocation(address, function(status, result) {

if (status === "complete" && result.info === 'OK') {

var geocode = result.geocodes[0];

rentMarker = new AMap.Marker({

map: map,

title: address,

icon: 'http://webapi.amap.com/theme/v1.3/markers/n/mark\_b.png',

position: [geocode.location.getLng(), geocode.location.getLat()]

});

rentMarkerArray.push(rentMarker);

rentMarker.content = "<div>房源：<a target = '\_blank' href='http://bj.58.com/pinpaigongyu/?key=" + address + "'>" + address + "</a><div>"

rentMarker.on('click', function(e) {

infoWindow.setContent(e.target.content);

infoWindow.open(map, e.target.getPosition());

if (amapTransfer) amapTransfer.clear();

amapTransfer = new AMap.Transfer({

map: map,

policy: AMap.TransferPolicy.LEAST\_TIME,

city: "北京市",

panel: 'transfer-panel'

});

amapTransfer.search([{

keyword: workAddress

}, {

keyword: address

}], function(status, result) {})

});

}

})

}

function delWorkLocation() {

if (polygonArray) map.remove(polygonArray);

if (workMarker) map.remove(workMarker);

polygonArray = [];

}

function delRentLocation() {

if (rentMarkerArray) map.remove(rentMarkerArray);

rentMarkerArray = [];

}

function loadWorkLocation() {

delWorkLocation();

var geocoder = new AMap.Geocoder({

city: "北京",

radius: 1000

});

geocoder.getLocation(workAddress, function(status, result) {

if (status === "complete" && result.info === 'OK') {

var geocode = result.geocodes[0];

x = geocode.location.getLng();

y = geocode.location.getLat();

loadWorkMarker(x, y);

loadWorkRange(x, y, 60, "#3f67a5", vehicle);

map.setZoomAndCenter(12, [x, y]);

}

})

}

function loadRentLocationByFile(fileName) {

delRentLocation();

var rent\_locations = new Set();

$.get(fileName, function(data) {

data = data.split("\n");

data.forEach(function(item, index) {

rent\_locations.add(item.split(",")[1]);

});

rent\_locations.forEach(function(element, index) {

addMarkerByAddress(element);

});

});

}

添加标尺，参考[带功能控件的地图](http://lbs.amap.com/api/javascript-api/example/map/map-with-function-control/" \t "_blank)。

var scale = new AMap.Scale();

map.addControl(scale);

一些需要放到全局的变量：

//公交到达圈对象

var arrivalRange = new AMap.ArrivalRange();

//经度，纬度，时间（用不到），通勤方式（默认是地铁＋公交）

var x, y, t, vehicle = "SUBWAY,BUS";

//工作地点，工作标记

var workAddress, workMarker;

//房源标记队列

var rentMarkerArray = [];

//多边形队列，存储公交到达的计算结果

var polygonArray = [];

//路径规划

var amapTransfer;

信息窗体的使用，在房源标记被点击时弹出，参考[给多个点添加信息窗体](http://lbs.amap.com/api/javascript-api/example/infowindow/add-infowindows-to-multiple-markers/" \t "_blank)。

//信息窗体对象

var infoWindow = new AMap.InfoWindow({

offset: new AMap.Pixel(0, -30)

});

//在房源标记被点击时打开

rentMarker.on('click', function(e) {

//鼠标移到标记上会显示标记content属性的内容

infoWindow.setContent(e.target.content);

//在标记的位置打开窗体

infoWindow.open(map, e.target.getPosition());

});

地址补完的使用，参考[输入提示后查询](http://lbs.amap.com/api/javascript-api/example/poi-search/search-after-enter-prompt/" \t "_blank)。

var auto = new AMap.Autocomplete({

//通过id指定输入元素

input: "work-location"

});

//添加事件监听，在选择补完的地址后调用workLocationSelected

AMap.event.addListener(auto, "select", workLocationSelected);

function workLocationSelected(e) {

//更新工作地点，加载公交到达圈

workAddress = e.poi.name;

loadWorkLocation();

}

loadWorkLocation的实现，这部分包含了地理编码的内容，参考[正向地理编码（地址-坐标）](http://lbs.amap.com/api/javascript-api/example/geocoder/geocoding/" \t "_blank)。

function loadWorkLocation() {

//首先清空地图上已有的到达圈

delWorkLocation();

var geocoder = new AMap.Geocoder({

city: "北京",

radius: 1000

});

geocoder.getLocation(workAddress, function(status, result) {

if (status === "complete" && result.info === 'OK') {

var geocode = result.geocodes[0];

x = geocode.location.getLng();

y = geocode.location.getLat();

//加载工作地点标记

loadWorkMarker(x, y);

//加载60分钟内工作地点到达圈

loadWorkRange(x, y, 60, "#3f67a5", vehicle);

//地图移动到工作地点的位置

map.setZoomAndCenter(12, [x, y]);

}

})

}

loadWorkRange的实现，在地图上绘制到达圈，参考：[公交到达圈](http://lbs.amap.com/api/javascript-api/example/bus-search/arrival-range/" \t "_blank)。

function loadWorkRange(x, y, t, color, v) {

arrivalRange.search([x, y], t, function(status, result) {

if (result.bounds) {

for (var i = 0; i < result.bounds.length; i++) {

//新建多边形对象

var polygon = new AMap.Polygon({

map: map,

fillColor: color,

fillOpacity: "0.4",

strokeColor: color,

strokeOpacity: "0.8",

strokeWeight: 1

});

//得到到达圈的多边形路径

polygon.setPath(result.bounds[i]);

polygonArray.push(polygon);

}

}

}, {

policy: v

});

}

载入房源信息功能的实现。由于安全问题，浏览器想要得到文件在系统内的位置就得用上一些技巧，这里还是算了，偷一下懒，因为房源文件跟index.html在同一个文件夹下，所以我们只要得到文件名就足够了。

function importRentInfo(fileInfo) {

var file = fileInfo.files[0].name;

loadRentLocationByFile(file);

}

我们使用一个集合来记录所有的房源地址。

function loadRentLocationByFile(fileName) {

//先删除现有的房源标记

delRentLocation();

//所有的地点都记录在集合中

var rent\_locations = new Set();

//jquery操作

$.get(fileName, function(data) {

data = data.split("\n");

data.forEach(function(item, index) {

rent\_locations.add(item.split(",")[1]);

});

rent\_locations.forEach(function(element, index) {

//加上房源标记

addMarkerByAddress(element);

});

});

}

addMarkerByAddress的实现参考：[按起终点名称规划路线](http://lbs.amap.com/api/javascript-api/example/bus-search/plan-route-according-to-name/" \t "_blank)与[点标记](http://lbs.amap.com/api/javascript-api/example/marker/marker-content/)。

注意其中这一句会被显示在信息窗体上。链接指向58品牌公寓馆的搜索页面，搜索的地址就是点标记（房源）的地址

rentMarker.content = "<div>房源：<a target = '\_blank' href='http://bj.58.com/pinpaigongyu/?key=" + address + "'>" + address + "</a><div>"

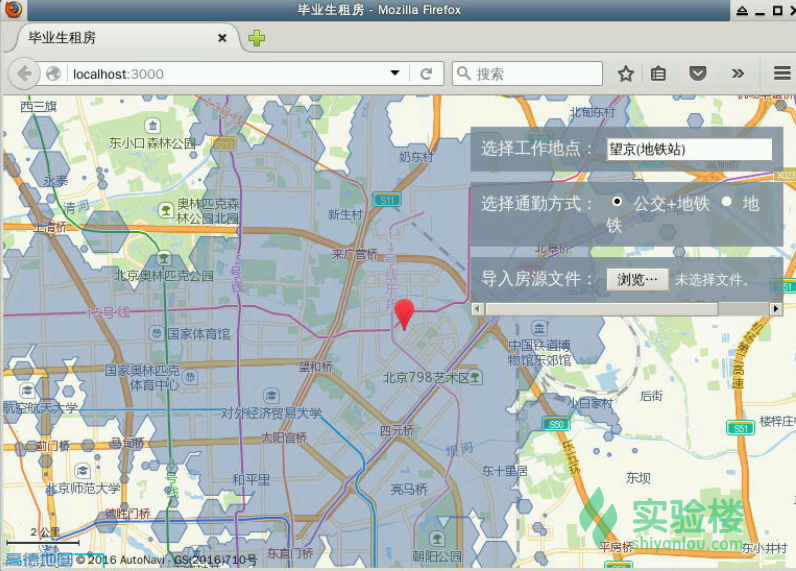
**5. 效果演示**

输入python -m SimpleHTTPServer 3000打开服务器，浏览器访问localhost:3000查看效果：

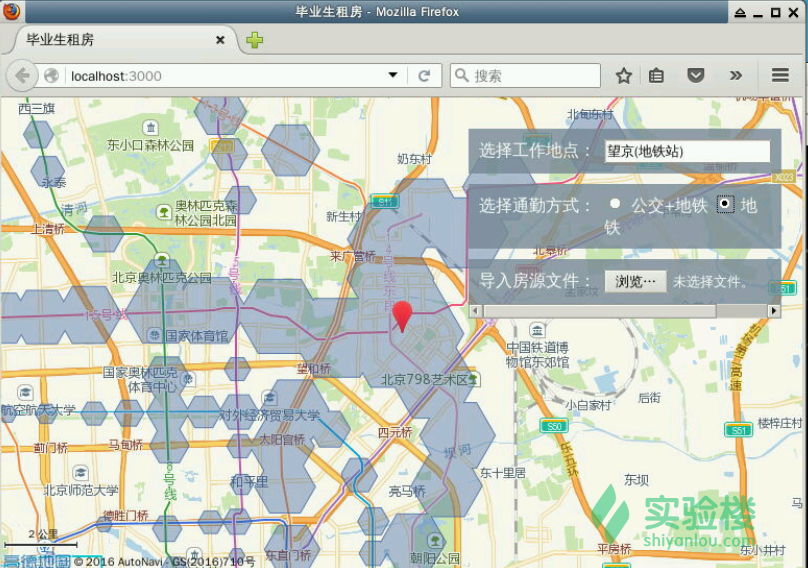
首先选择工作地点：



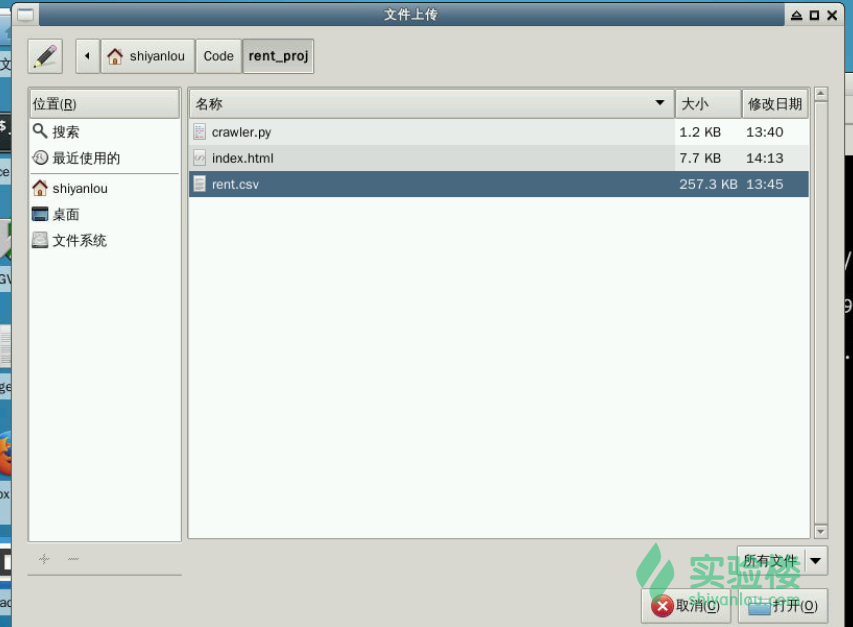
划出了一小时内的通勤范围：



北京堵车太猖狂，可能还是地铁保险：



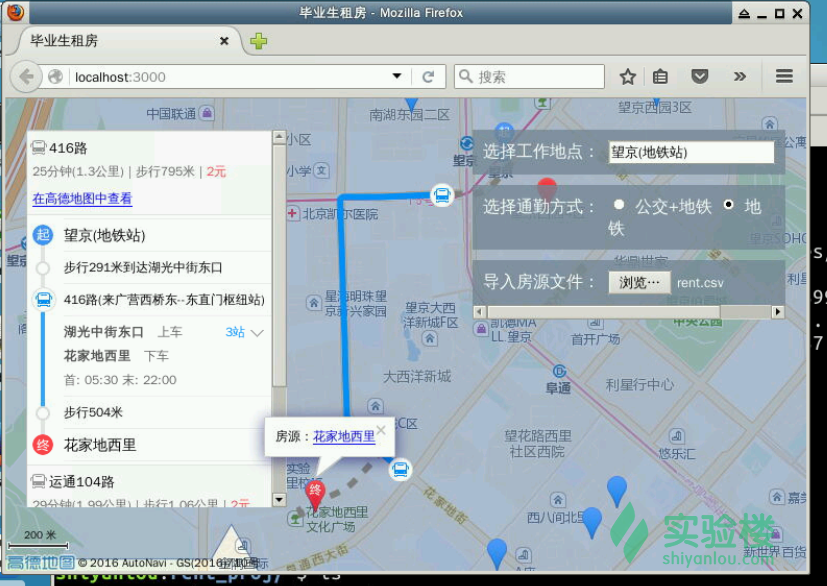
导入房源文件：



导入后：



选择一处房源，会自动帮你规划路径：



选中房源地址跳转到目标页面：



**五、总结**

多多利用网上的各种开放平台与API能让你的编程事半功倍，甚至这可能改变一个人的生活，影响到一个人的一生也说不定。这种说法听起来似乎太夸张，但是就拿高考报志愿举例吧，选什么学校，什么专业，可能就是一个决策导致了人生的截然不同。信息不对等或是缺乏对信息的分析都是很吃亏的。互联网时代，信息唾手可得，但是呢，我花大把时间看到的都不是我想要的，即使在看个人房源还是得自己排除一堆中介。我不是常常出门所以对交通不熟，把这些信息都查一遍又得花大量的时间。还好现在有了各种好的平台与开放接口，只需要你有一点点编程技能和一个想要实现什么东西的想法，一切就都不一样了。

这个项目有很多可以改进的部分，比如房源信息的选取过于简单，房间的面积也应该纳入考虑。比如可以做成在地图上显示具体的房源信息，每个标记都只标记唯一的房源。比如房源信息为什么要存文件呢，也可以存数据库里。比如爬取房源的效率太低，可以改用scrapy或者以自己的方式进行优化。比如只爬取一个网站会不会房源太少，也可以多找几处，把豆瓣的租房贴，各个大学的论坛也算进去，也许能真正解决租房难的问题呢。