目录

正则表达式应用举例

- 最新版本： `v1.0`
- 更新时间： `20200202`

鸣谢

感谢我的老婆陈雪雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

简介

整理正则表达式各个领域的应用，并给出实际使用案例，供参考

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/regex_usage_examples: 正则表达式应用举例

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 正则表达式应用举例 book.crifan.com
- 正则表达式应用举例 crifan.github.io

离线下载阅读

- 正则表达式应用举例 PDF
- 正则表达式应用举例 ePub
- 正则表达式应用举例 Mobi

版权说明

正则应用概述

如 应用广泛的超强搜索：正则表达式所说，正则的用途非常广泛，下面来整理各个领域内正则的用途和具体例子。

目的在于：

- 没听过正则的人：了解正则在搜索和替换等方面的功能是多么强大
- 听过正则但不熟悉的人：能大概了解不同领域内，正则大概是怎么使用的
- 需要使用正则的人：借鉴正则的写法，在需要的时候，自己写正则，满足自己的需求

正则应用举例

下面来介绍正则表达式在不同领域的各种应用。

JavaScript

代码：

```
    let cowCodeRegex = new RegExp("/cowActivity/([^/]+)", "g");
    console.log(`cowCodeRegex=${cowCodeRegex}`);
    let foundCowCode = cowCodeRegex.exec(this.state.curUrl); ///uapp/cowActivity/12-0985/1
/1522936800000
    console.log(`foundCowCode=${foundCowCode}`); ///cowActivity/13-6234,13-6234
    let cowCode = null;
    if (foundCowCode) {
      let inputStr = foundCowCode[0];
      console.log(`inputStr=${inputStr}`); ///uapp/cowActivity/11-5953
      cowCode = foundCowCode[1];
      console.log(`cowCode=${cowCode}`); //11-5953
    }
```

输出：

```
parsePageType: currentUrl /uapp/cowActivity/12-0985/1/1522936800000 -> page 牛只活动量
index.js?5c2a:528 cowCodeRegex /\/cowActivity\/([^\/]+)/g
index.js?5c2a:530 foundCowCode /cowActivity/12-0985,12-0985
index.js?5c2a:534 inputStr /cowActivity/12-0985
index.js?5c2a:536 cowCode 12-0985
index.js?5c2a:540 fullTitle 牛只活动量 12-0985
```

Python

BeautifulSoup

`BeautifulSoup` 中的 `find` 和 `findAll` 的 `name` 或 `attr` 参数，支持正则写法：

代码：

```
h1userSoupList = soup.findAll(name="h1", attrs={"class":re.compile(r"h1user(\s\w+)?")});
```

可以从html：

```
<div class="icon_col">
        <h1 class="h1user">crifan</h1>
        <h1 class="h1user test1">crifan 123</h1>
        <h1 class="h1user test2">crifan 456</h1>
</div>
```

搜到列表：

```
class="h1user"
class="h1user test1"
class="h1user test2"
```

数据库

MongoDB

相关代码：

```
if args.get('title'):
    filters['title'] = {'$regex': args['title'], '$options': 'i'}

if args.get('unitCode'):
    filters['unit_code'] = {'$regex': args['unitCode'], '$options': 'i'}
```

其中的 `$regex` 表示搜索时，用正则搜索

-》此处含义是：去搜索 `title` 和 `unitCode` 两个字段，且 `i` 表示不区分大小写

MongoDB Compass

`MongoDB Compass` 中想要用正则搜索字段：

```
grading
    lexile: "AD450L"
```

写法是：

```
{"grading.lexile": {$regex: "AD.*"}}
```

或：regex加上行首和行尾判断：

```
{"grading.lexile": {$regex: "^AD.*$"}}
```

或 regex用引号引起来

```
{"grading.lexile": {"$regex": "AD.*"}}
```

详见：

【整理Book】主流文档型数据库：MongoDB

编辑器和IDE

Sublime

提取youtube网页返回的html中包含的子页面的url地址

比如，用：

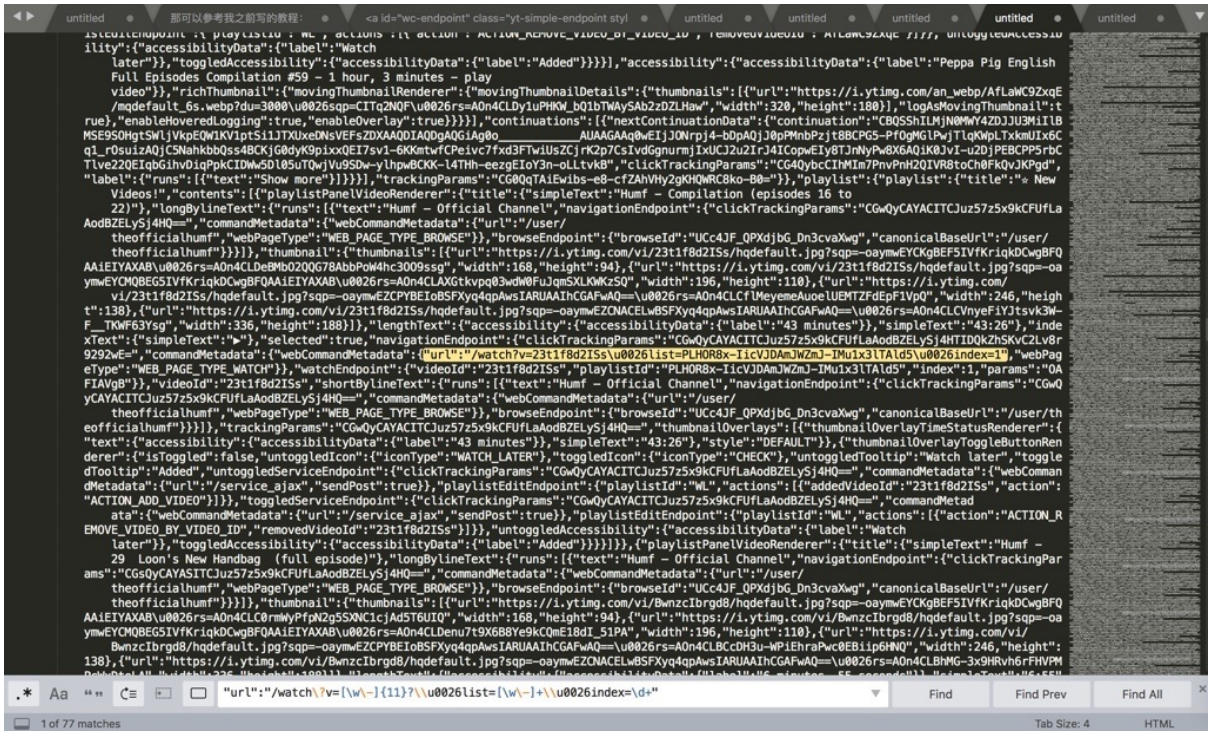
```
"url":"/watch\?v=[\w\-]{11}?\\u0026list=[\w\-]+\\u0026index=\d+"
```

可以从一堆的js中的script的值中：

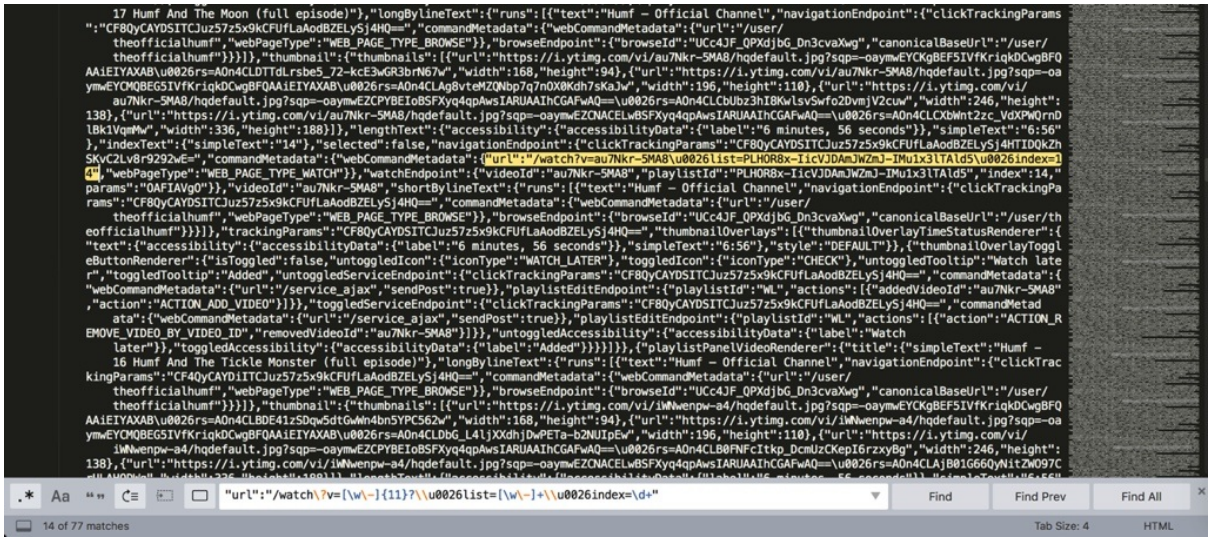
```
{"webCommandMetadata":{"url":"/watch?v=23t1f8d2ISs\u0026list=PLHOR8x-IicVJDAmJWZmJ-IMu1x3lTAld5\u0026index=1",”webPageType"

{"webCommandMetadata":{"url":"/watch?v=au7Nkr-5MA8\u0026list=PLHOR8x-IicVJDAmJWZmJ-IMu1x3lTAld5\u0026index=14",”webPageType"
```

搜索出所要的内容，此处有77个符合需要的内容： `1/77`

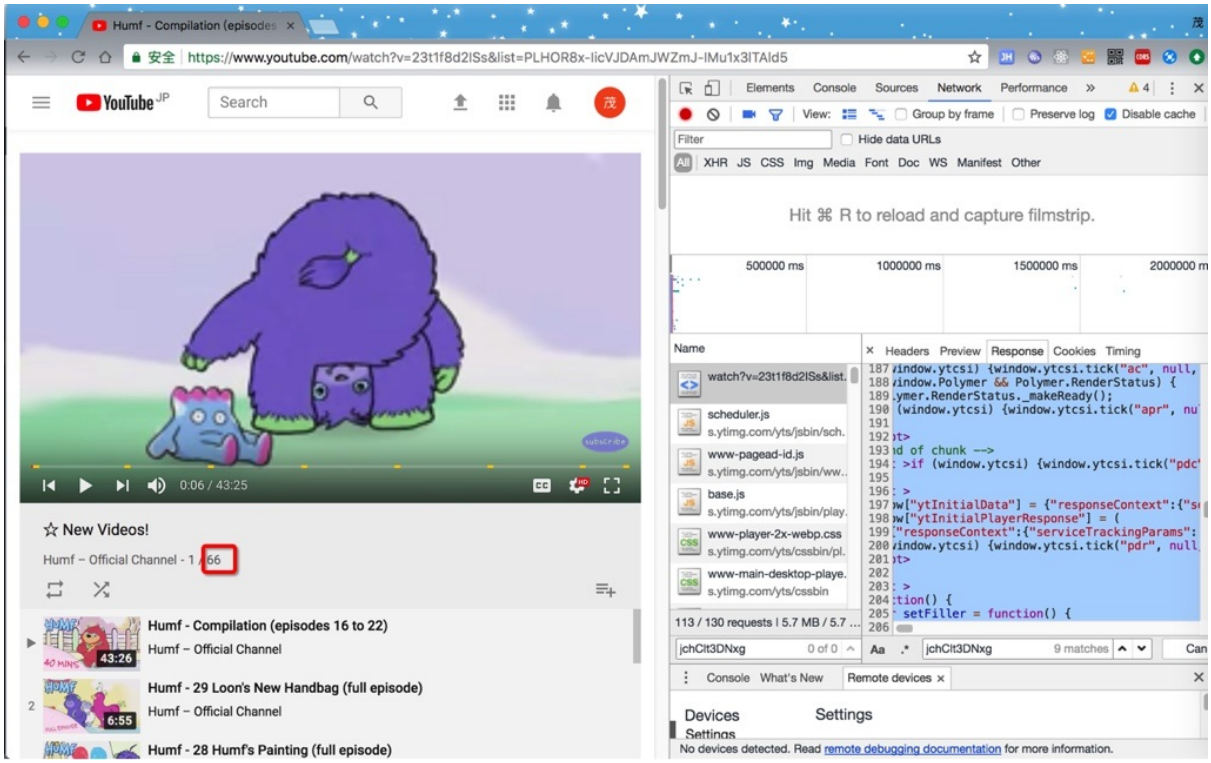


点击Find，继续向下找，比如找到第14个： `14/77`



另外，进一步的举例：

此处，对应着页面上的其实只是希望找到66个地址就可以了：



但是此处找到77个，多出11个，则是由于：

此处的js的变量的值中，包含了不需要的额外的11个

所以此时，由于没法方便的从字符串中区别开来，不好去掉另外那11个，则只能：

想办法拿到js的变量值，然后通过转换为json，然后再去获取json对象中的值，即可准确的得到所需要的值。

所以此处，又可以接着通过正则去先得到js的变量的值：

用正则：

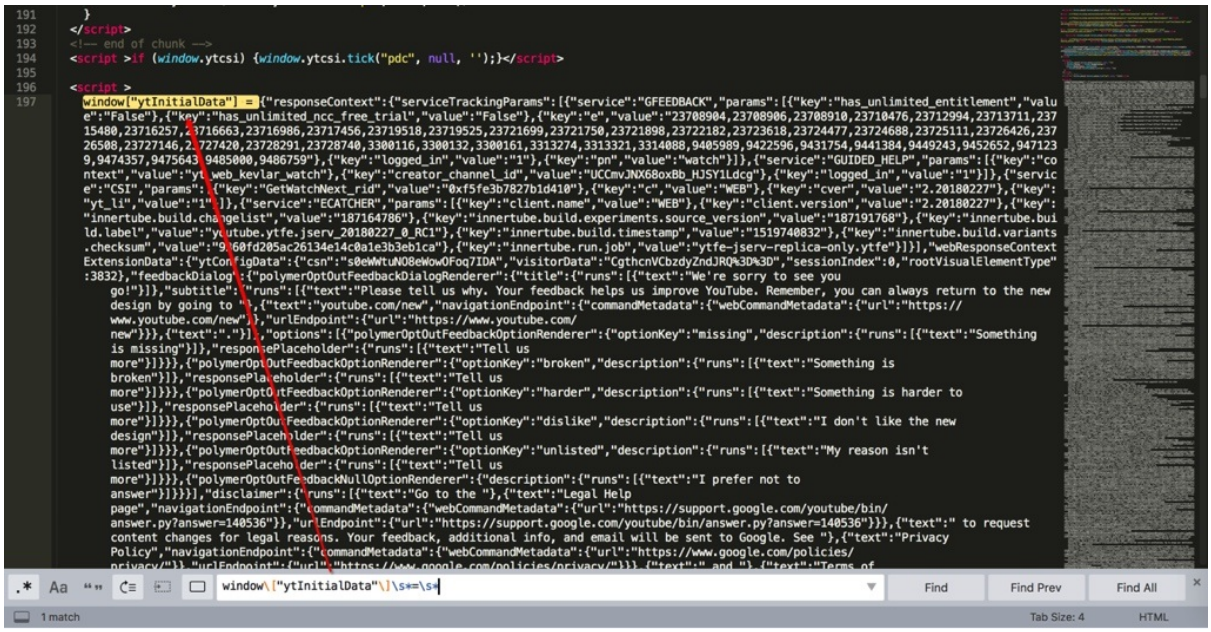
```
window\["ytInitialData"\]\s* \s*

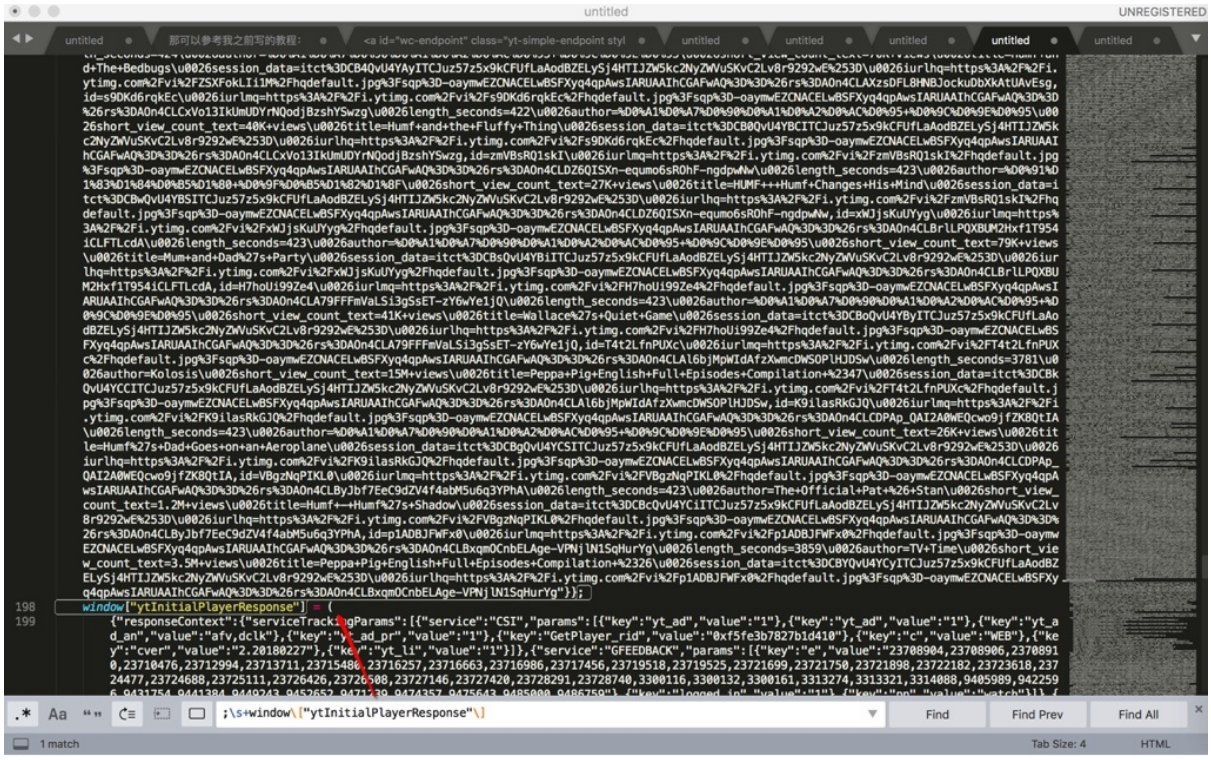
;\s+window\["ytInitialPlayerResponse"\]
```

从：

```
  <script >
    window["ytInitialData"] = {"responseContext":
...
xqmOCnbELAge_VPNjlN1SqHurYg"}};
    window["ytInitialPlayerResponse"] = (
```

中，搜索到所要的内容：



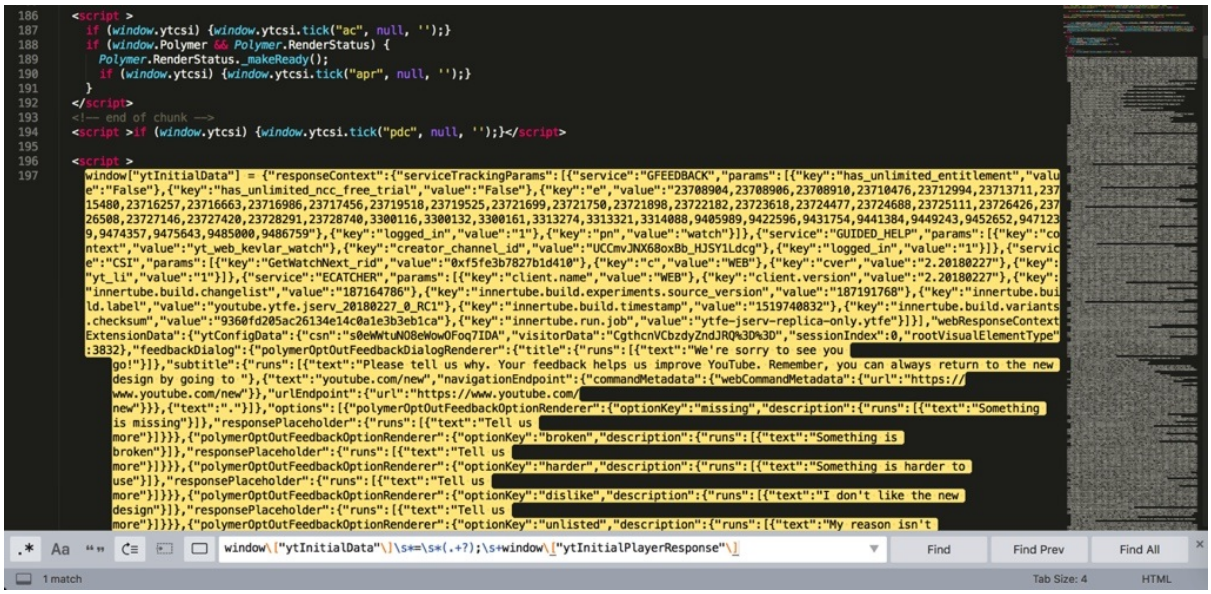


而首尾的正则之间的内容，就是需要找的js的变量的值，是个json

对应着写个完整的正则：

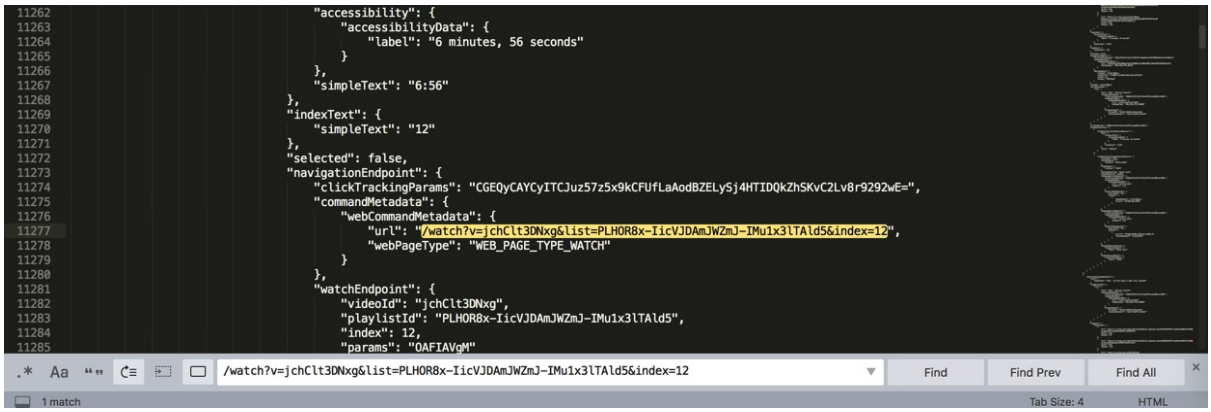
```
window\["ytInitialData"\]\s*=\s*(.+?);\s+window\["ytInitialPlayerResponse"\]
```

就可以匹配到这段完整的内容了：



后续就可以通过解析json去精确获取所要的url的值了。

比如第12个url：



由此实现了：

根据自己的实际的（业务）需求，通过充分利用正则表达式，获取想要的符合特定某一规则的内容。

html中提取出浙江省的每个市到Xmind中

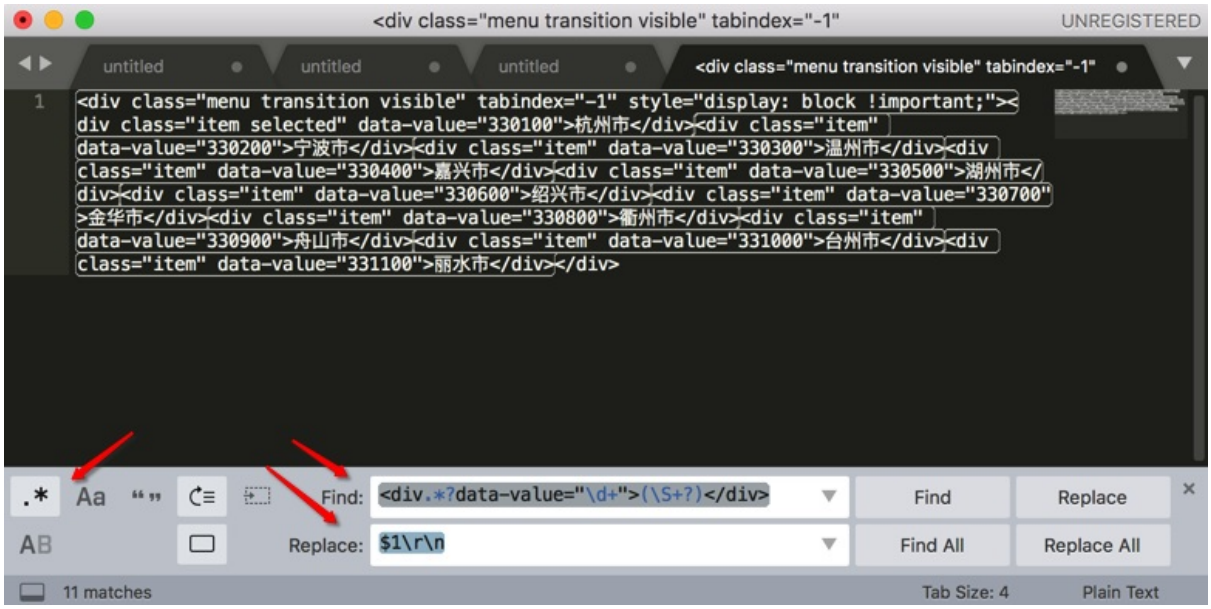
```
<div class="menu transition visible" tabindex="-1" style="display: block !important;"><div
class="item selected" data-value="330100">杭州市</div><div class="item" data-value="330200">
宁波市</div><div class="item" data-value="330300">温州市</div><div class="item" data-value="
330400">嘉兴市</div><div class="item" data-value="330500">湖州市</div><div class="item" data
-value="330600">绍兴市</div><div class="item" data-value="330700">金华市</div><div class="it
em" data-value="330800">衢州市</div><div class="item" data-value="330900">舟山市</div><div c
lass="item" data-value="331000">台州市</div><div class="item" data-value="331100">丽水市</div
></div>
```

希望提取出每个市

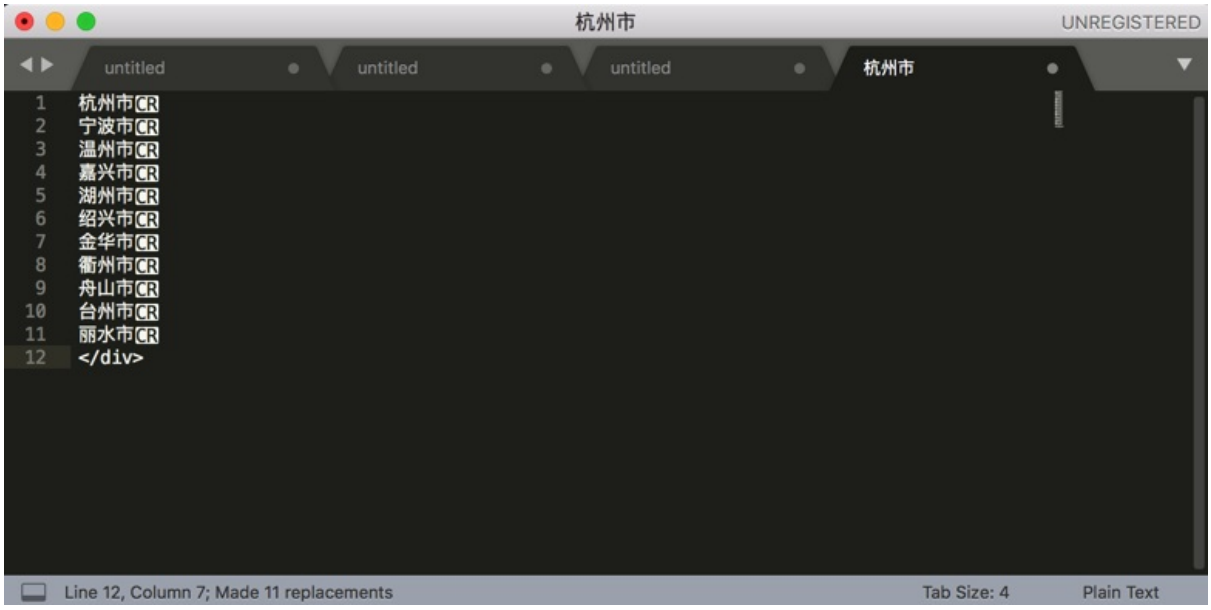
正则写法：

- 查找Find： `<div.*?data-value="\d+">(\S+?)</div>`
- 替换Replace： `$1\r\n`

点击 `Replace All` ：



替换成：



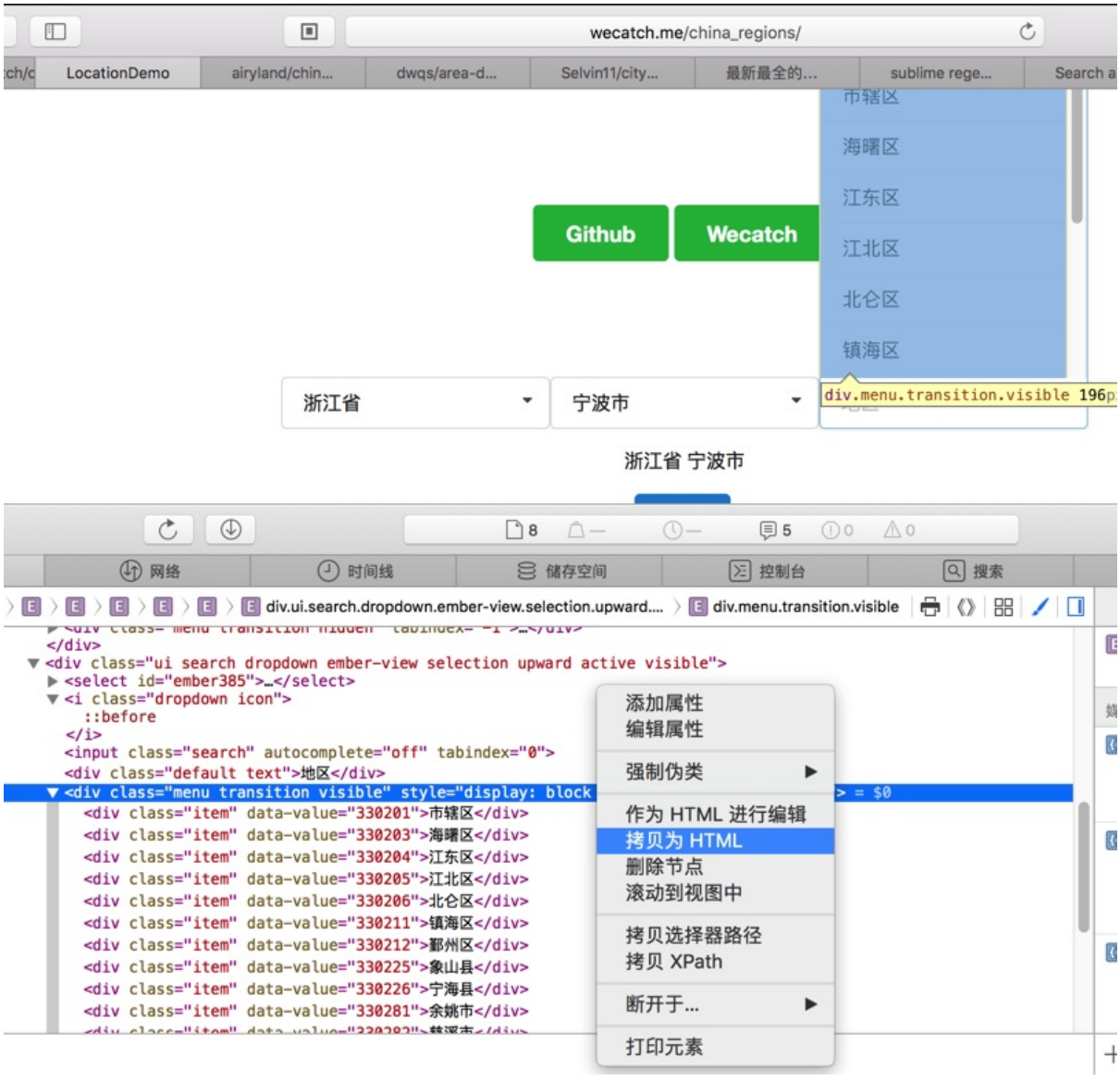
忽略掉最后的 `</div>` ，拷贝出来，即可得到我要的所有的市：

```
杭州市
宁波市
温州市
嘉兴市
湖州市
```

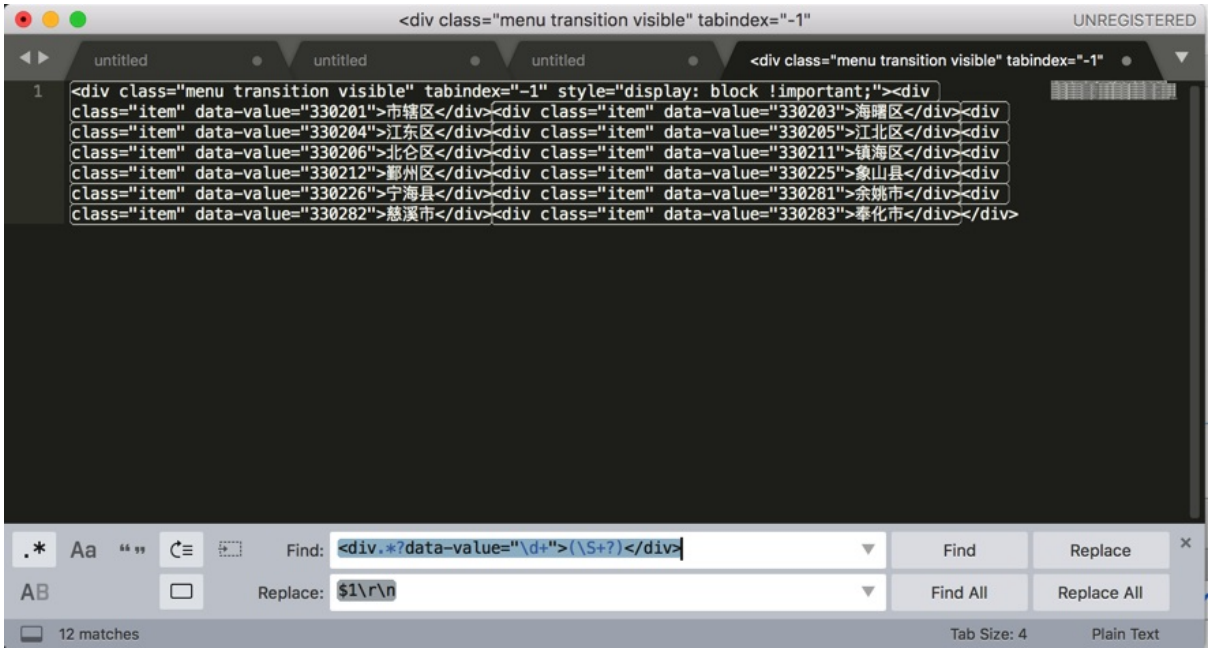
```
绍兴市
金华市
衢州市
舟山市
台州市
丽水市
```

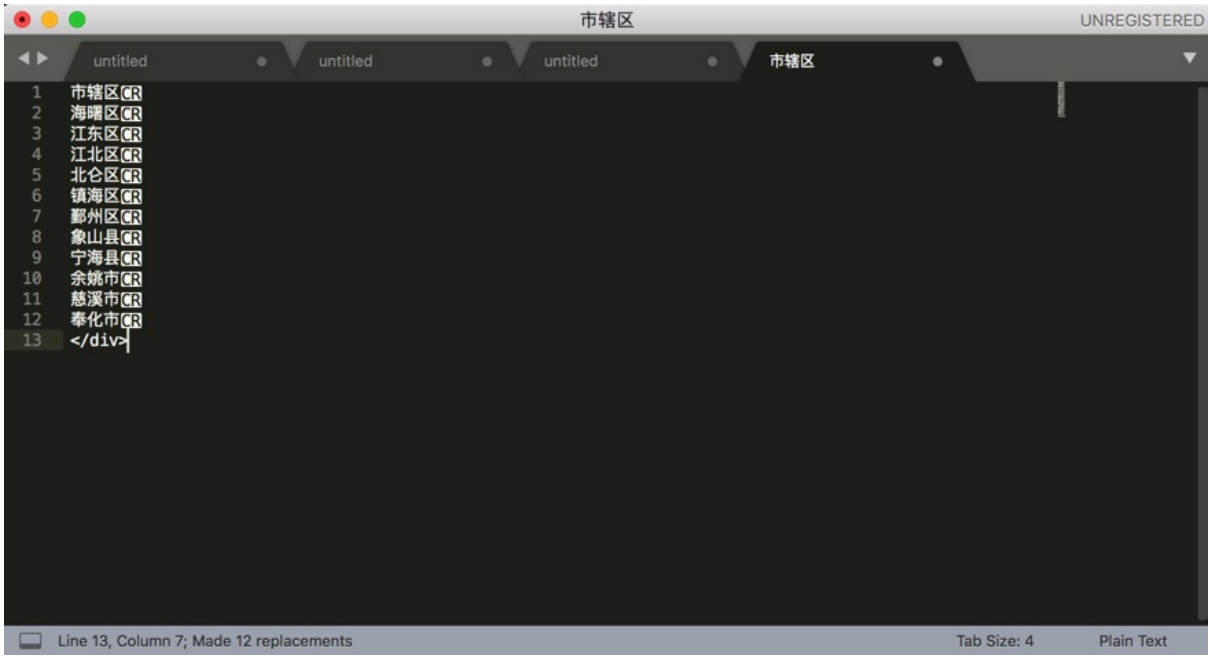
->

如此继续重复此步骤，直到把网页中的内容：

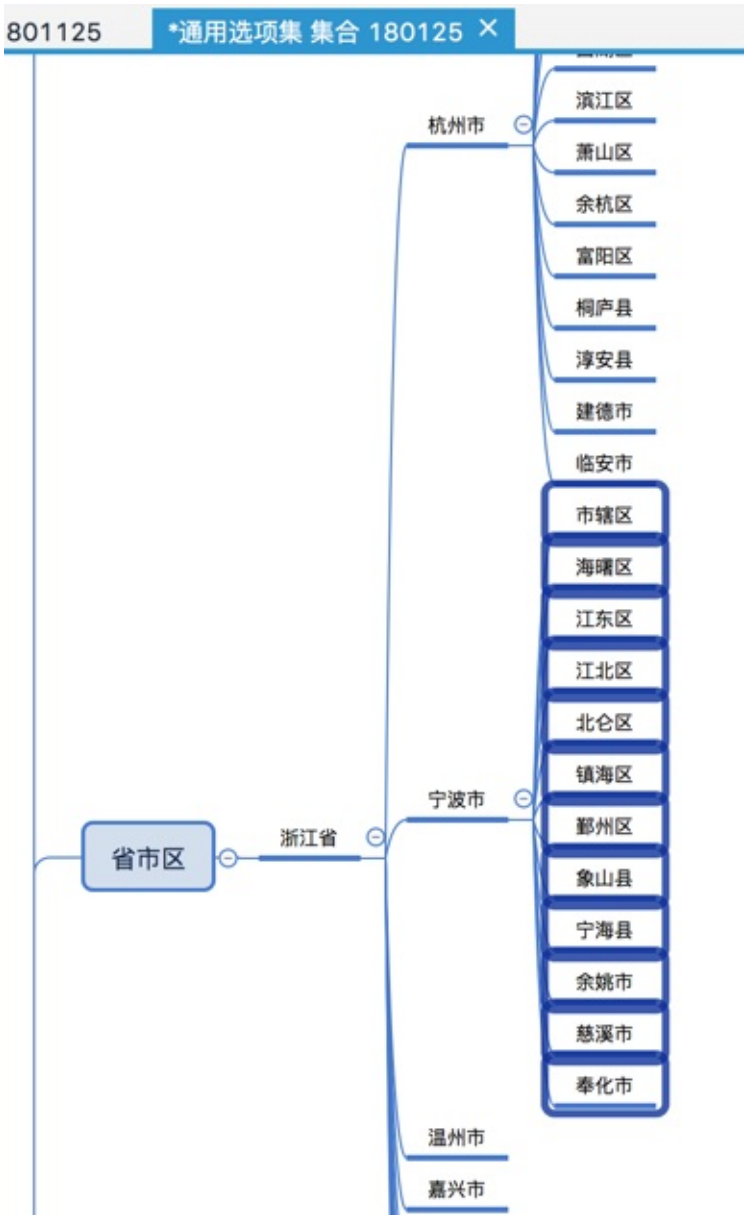


分多次，但是是批量的：





全部都整理到Xmind中：



就不用一个个拷贝，一个个粘贴了 -》 从而提高工作效率。

VSCode

普通的搜索和替换

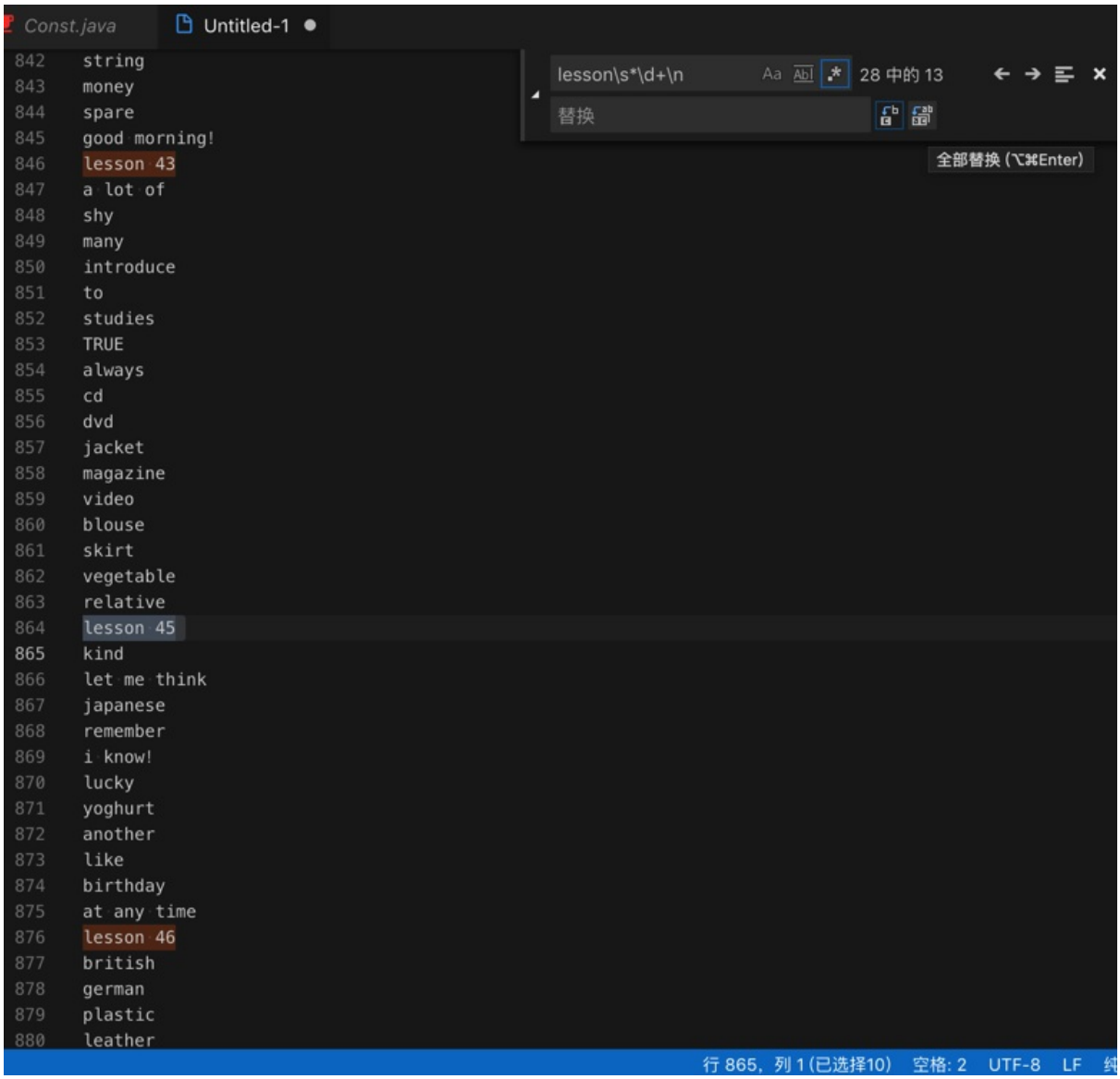
下面是一些相对普通的正则的搜索和替换的应用举例：

去除内容中多余的 lessionxxx的单词

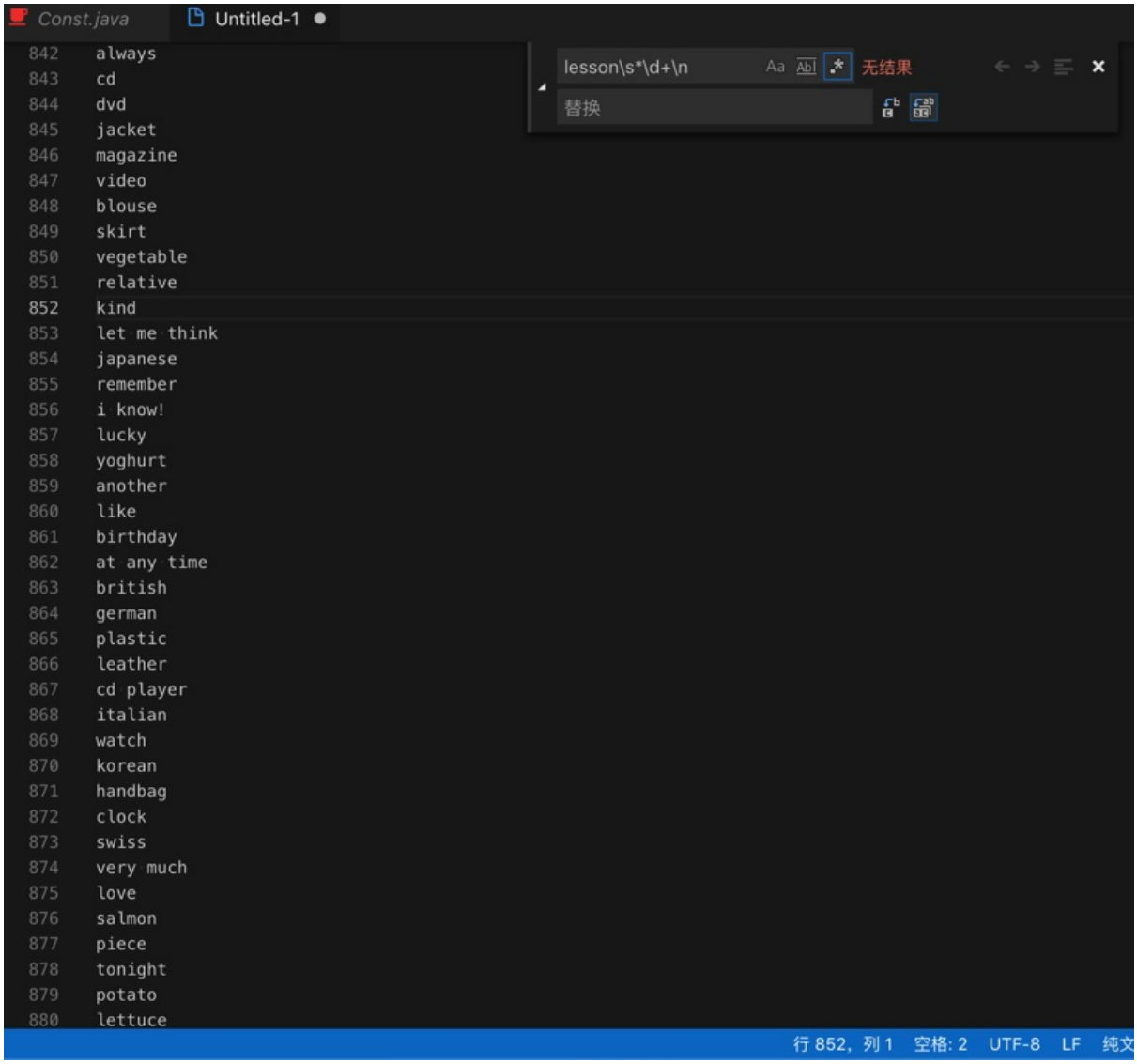
正则：

```
lesson\s*\d+\n
```

从：



替换成：

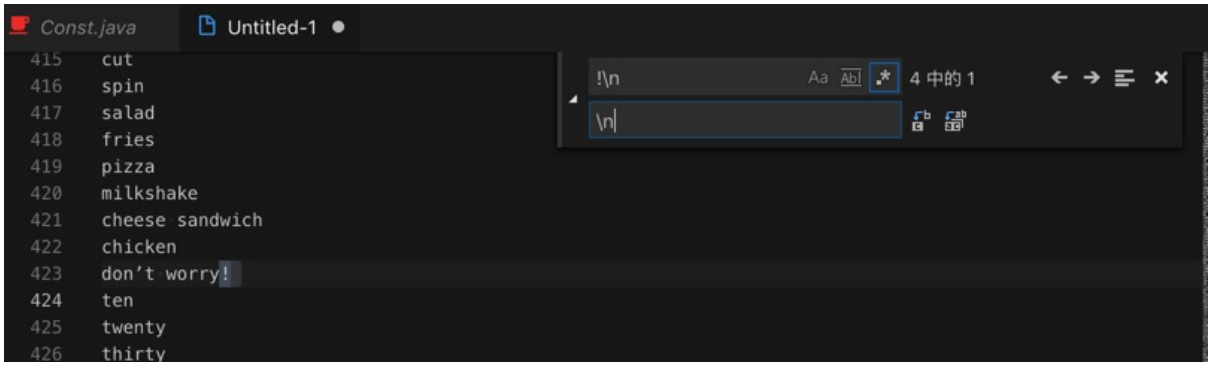


语句末尾去掉感叹号

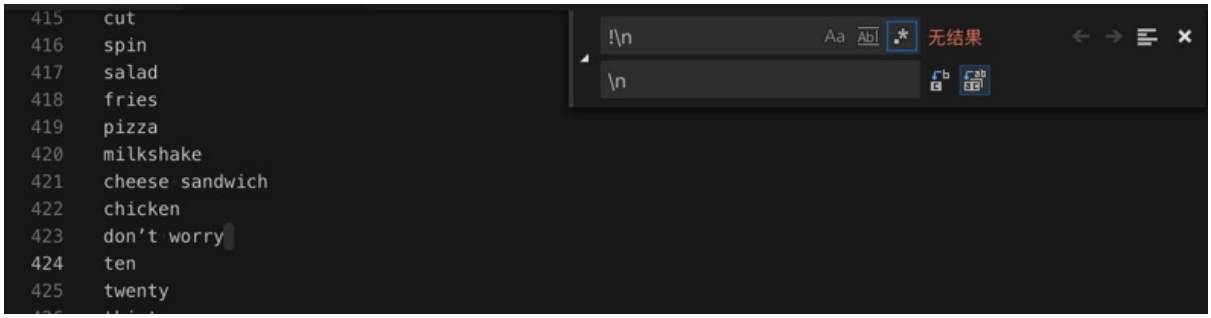
正则：

```
 \n
\n
```

从：



替换成：



高级的搜索和替换

下面是一些，相对高级一些的用法，比如：

- 搜索带 `分组`
  - 即：`(xxx)`
- 替换时带 `引用`
  - 即：`$N`
    - N=1,2,3,...

等正则应用举例，供参考。

文章标题和链接转换为Markdown的链接

正则替换规则：

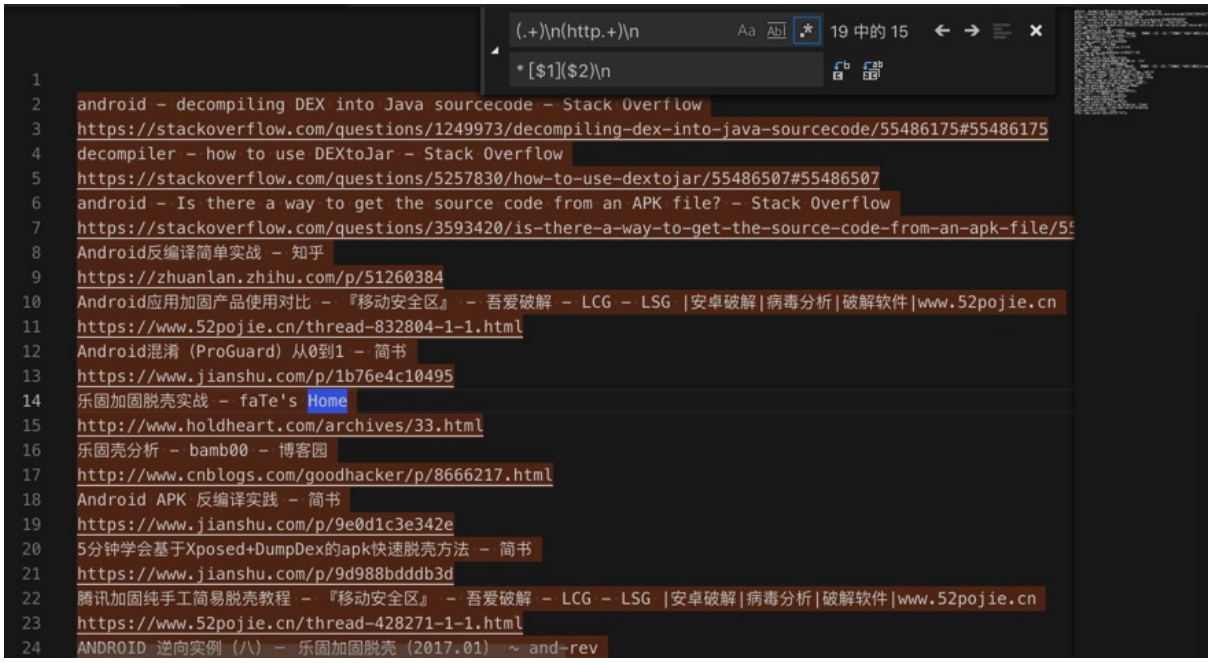
```
(.+)\n(http.+)\n
* [$1]($2)\n
```

从：

```
android - decompiling DEX into Java sourcecode - Stack Overflow
https://stackoverflow.com/questions/1249973/decompiling-dex-into-java-sourcecode/55486175#
55486175
decompiler - how to use DEXtoJar - Stack Overflow
https://stackoverflow.com/questions/5257830/how-to-use-dextojar/55486507#55486507
android - Is there a way to get the source code from an APK file? - Stack Overflow
```

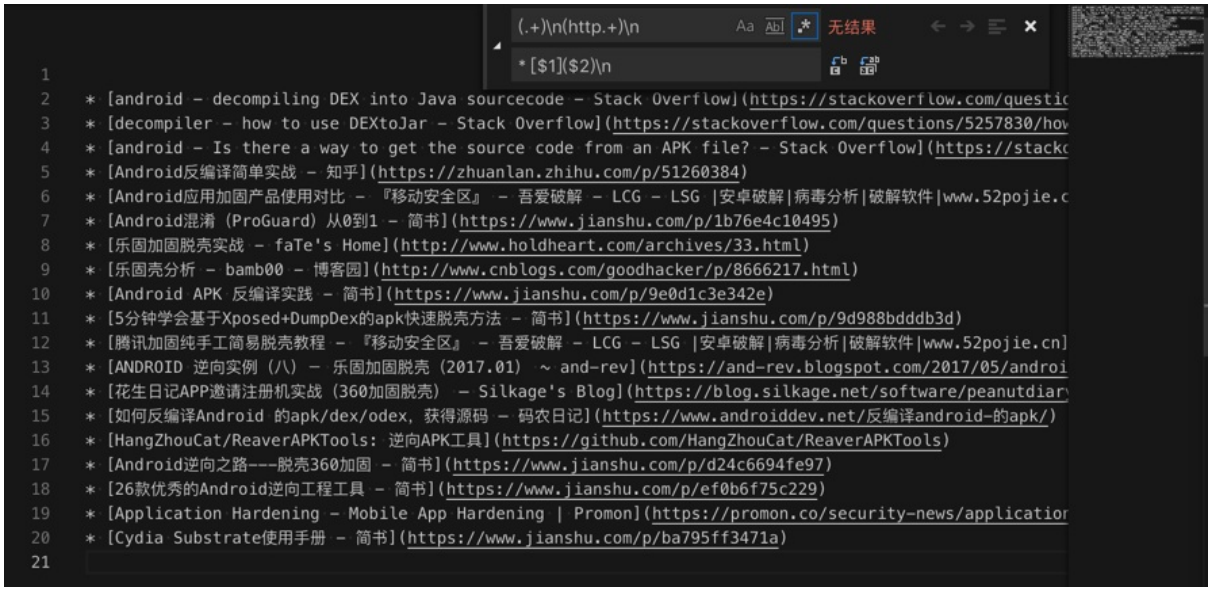
```
https://stackoverflow.com/questions/3593420/is-there-a-way-to-get-the-source-code-from-an-apk-file/55567538#55567538
Android反编译简单实战 - 知乎
https://zhuanlan.zhihu.com/p/51260384
Android应用加固产品使用对比 - 『移动安全区』 - 吾爱破解 - LCG - LSG 安卓破解 病毒分析 破解软件 www.52pojie.cn
https://www.52pojie.cn/thread-832804-1-1.html
Android混淆（ProGuard）从0到1 - 简书
https://www.jianshu.com/p/1b76e4c10495
乐固加固脱壳实战 - faTe's Home
http://www.holdheart.com/archives/33.html
乐固壳分析 - bamb00 - 博客园
http://www.cnblogs.com/goodhacker/p/8666217.html
Android APK 反编译实践 - 简书
https://www.jianshu.com/p/9e0d1c3e342e
5分钟学会基于Xposed+DumpDex的apk快速脱壳方法 - 简书
https://www.jianshu.com/p/9d988bdddb3d
腾讯加固纯手工简易脱壳教程 - 『移动安全区』 - 吾爱破解 - LCG - LSG |安卓破解|病毒分析|破解软件|www.52pojie.cn
https://www.52pojie.cn/thread-428271-1-1.html
ANDROID 逆向实例（八） — 乐固加固脱壳（2017.01） ~ and-rev
https://and-rev.blogspot.com/2017/05/android-201701.html
花生日记APP邀请注册机实战（360加固脱壳） - Silkage's Blog
https://blog.silkage.net/software/peanutdiary.html
如何反编译Android 的apk/dex/odex，获得源码 – 码农日记
https://www.androiddev.net/反编译android-的apk/
HangZhouCat/ReaverAPKTools: 逆向APK工具
https://github.com/HangZhouCat/ReaverAPKTools
Android逆向之路---脱壳360加固 - 简书
https://www.jianshu.com/p/d24c6694fe97
26款优秀的Android逆向工程工具 - 简书
https://www.jianshu.com/p/ef0b6f75c229
Application Hardening - Mobile App Hardening | Promon
https://promon.co/security-news/application-hardening/
Cydia Substrate使用手册 - 简书
https://www.jianshu.com/p/ba795ff3471a
```

把：



换成：

```
* [android - decompiling DEX into Java sourcecode - Stack Overflow](https://stackoverflow.com/questions/1249973/decompiling-dex-into-java-sourcecode/55486175#55486175)
* [decompiler - how to use DEXtoJar - Stack Overflow](https://stackoverflow.com/questions/5257830/how-to-use-dextojar/55486507#55486507)
* [android - Is there a way to get the source code from an APK file? - Stack Overflow](https://stackoverflow.com/questions/3593420/is-there-a-way-to-get-the-source-code-from-an-apk-file/55567538#55567538)
* [Android反编译简单实战 - 知乎](https://zhuanlan.zhihu.com/p/51260384)
* [Android应用加固产品使用对比 - 『移动安全区』 - 吾爱破解 - LCG - LSG |安卓破解|病毒分析|破解软件|www.52pojie.cn](https://www.52pojie.cn/thread-832804-1-1.html)
* [Android混淆（ProGuard）从0到1 - 简书](https://www.jianshu.com/p/1b76e4c10495)
* [乐固加固脱壳实战 - faTe's Home](http://www.holdheart.com/archives/33.html)
* [乐固壳分析 - bamb00 - 博客园](http://www.cnblogs.com/goodhacker/p/8666217.html)
* [Android APK 反编译实践 - 简书](https://www.jianshu.com/p/9e0d1c3e342e)
* [5分钟学会基于Xposed+DumpDex的apk快速脱壳方法 - 简书](https://www.jianshu.com/p/9d988bdddb3d)
* [腾讯加固纯手工简易脱壳教程 - 『移动安全区』 - 吾爱破解 - LCG - LSG |安卓破解|病毒分析|破解软件|www.52pojie.cn](https://www.52pojie.cn/thread-428271-1-1.html)
* [ANDROID 逆向实例（八） — 乐固加固脱壳（2017.01） ~ and-rev](https://and-rev.blogspot.com/2017/05/android-201701.html)
* [花生日记APP邀请注册机实战（360加固脱壳） - Silkage's Blog](https://blog.silkage.net/software/peanutdiary.html)
* [如何反编译Android 的apk/dex/odex，获得源码 - 码农日记](https://www.androiddev.net/反编译android-的apk/)
* [HangZhouCat/ReaverAPKTools: 逆向APK工具](https://github.com/HangZhouCat/ReaverAPKTools)
* [Android逆向之路---脱壳360加固 - 简书](https://www.jianshu.com/p/d24c6694fe97)
* [26款优秀的Android逆向工程工具 - 简书](https://www.jianshu.com/p/ef0b6f75c229)
* [Application Hardening - Mobile App Hardening | Promon](https://promon.co/security-news/application-hardening/)
* [Cydia Substrate使用手册 - 简书](https://www.jianshu.com/p/ba795ff3471a)
```



用于：放在 `markdown` 作为参考资料。

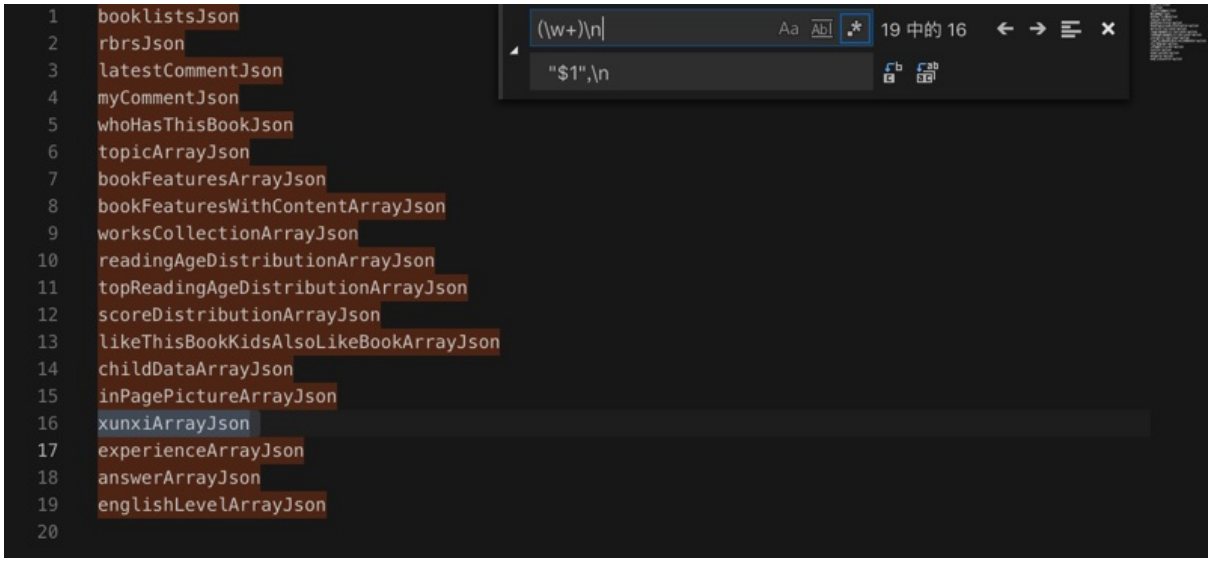
json后缀的字符串变成代码中字符串列表

正则：

```
(\w+)\n
  "$1",\n
```

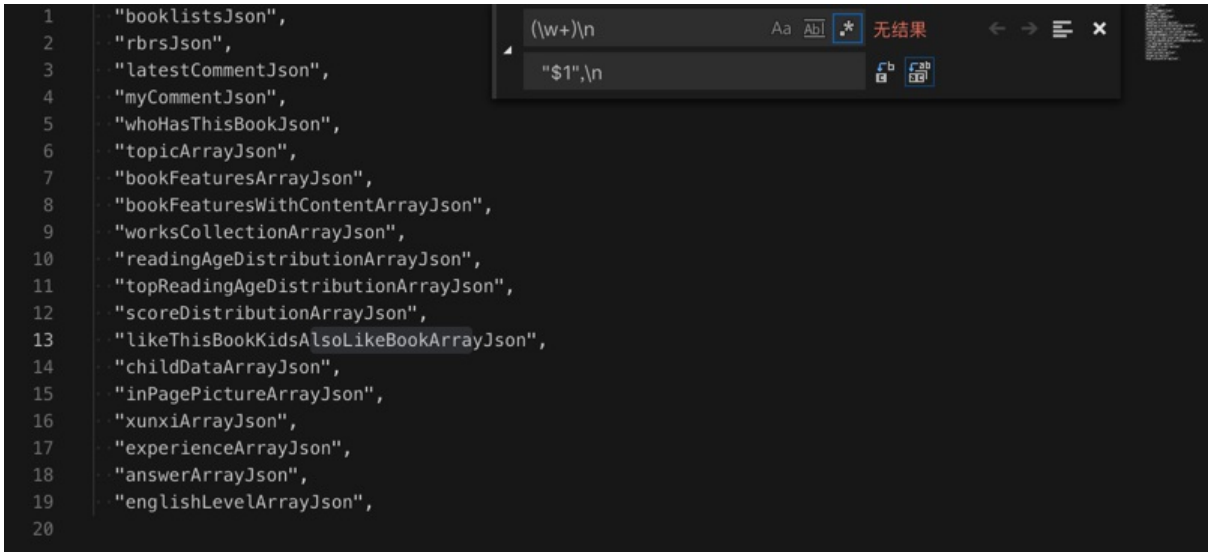
从：

```
booklistsJson
rbrsJson
latestCommentJson
myCommentJson
whoHasThisBookJson
topicArrayJson
bookFeaturesArrayJson
bookFeaturesWithContentArrayJson
worksCollectionArrayJson
readingAgeDistributionArrayJson
topReadingAgeDistributionArrayJson
scoreDistributionArrayJson
likeThisBookKidsAlsoLikeBookArrayJson
childDataArrayJson
inPagePictureArrayJson
xunxiArrayJson
experienceArrayJson
answerArrayJson
englishLevelArrayJson
```



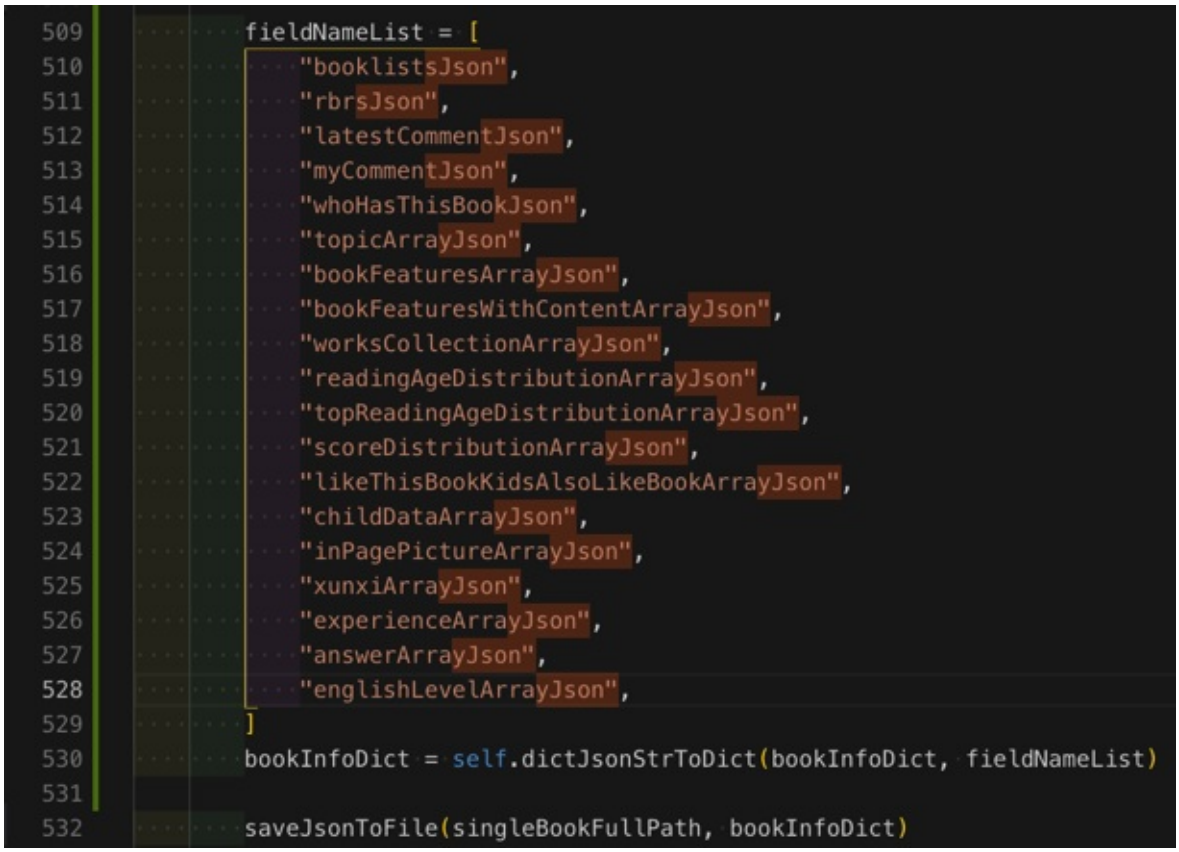
变成：

```
"booklistsJson",
"rbrsJson",
"latestCommentJson",
"myCommentJson",
"whoHasThisBookJson",
"topicArrayJson",
"bookFeaturesArrayJson",
"bookFeaturesWithContentArrayJson",
"worksCollectionArrayJson",
"readingAgeDistributionArrayJson",
"topReadingAgeDistributionArrayJson",
"scoreDistributionArrayJson",
"likeThisBookKidsAlsoLikeBookArrayJson",
"childDataArrayJson",
"inPagePictureArrayJson",
"xunxiArrayJson",
"experienceArrayJson",
"answerArrayJson",
"englishLevelArrayJson",
```



用于：

拷贝到代码里，用于列表变量的值：



省去：自己手动去对每一行手动去加上 `""` 再控制 `缩进` 的繁琐工作了。

获取到康美通的版本历史

正则：

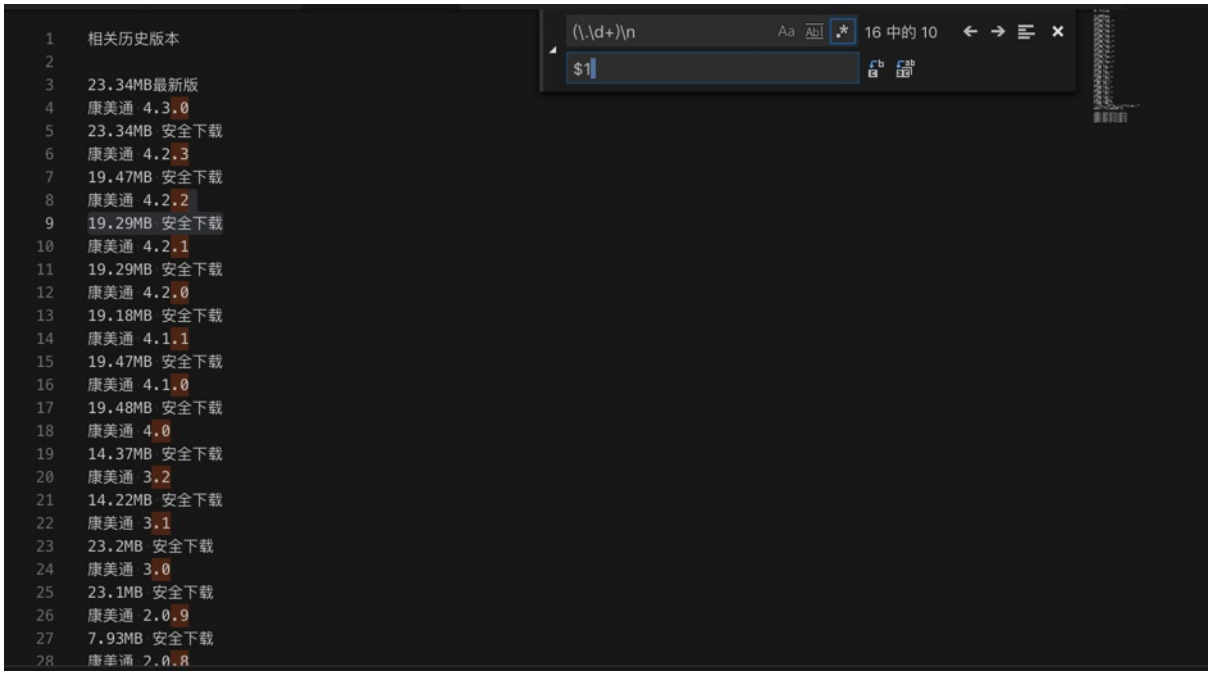
```
(\.\d+)\n
```

```
$1
```

从：

```
相关历史版本


23.34MB最新版
康美通 4.3.0
23.34MB 安全下載
康美通 4.2.3
19.47MB 安全下載
康美通 4.2.2
19.29MB 安全下載
康美通 4.2.1
19.29MB 安全下載
康美通 4.2.0
19.18MB 安全下載
康美通 4.1.1
19.47MB 安全下載
康美通 4.1.0
19.48MB 安全下載
康美通 4.0
14.37MB 安全下載
康美通 3.2
14.22MB 安全下載
康美通 3.1
23.2MB 安全下載
康美通 3.0
23.1MB 安全下載
康美通 2.0.9
7.93MB 安全下載
康美通 2.0.8
7.3MB 安全下載
康美通 2.0.7
7.29MB 安全下載
康美通 2.0.6
7.29MB 安全下載
康美通 1.0.1
6.27MB 安全下載
康美通 1.0beta
5.86MB 安全下載
 下載豌豆荚客户端（更多历史版本）下載
康美通 历史版本年份合集
```



替换成：

相关历史版本

23.34MB最新版
康美通 4.3.0 23.34MB 安全下载
康美通 4.2.3 19.47MB 安全下载
康美通 4.2.2 19.29MB 安全下载
康美通 4.2.1 19.29MB 安全下载
康美通 4.2.0 19.18MB 安全下载
康美通 4.1.1 19.47MB 安全下载
康美通 4.1.0 19.48MB 安全下载
康美通 4.0 14.37MB 安全下载
康美通 3.2 14.22MB 安全下载
康美通 3.1 23.2MB 安全下载
康美通 3.0 23.1MB 安全下载
康美通 2.0.9 7.93MB 安全下载
康美通 2.0.8 7.3MB 安全下载
康美通 2.0.7 7.29MB 安全下载
康美通 2.0.6 7.29MB 安全下载
康美通 1.0.1 6.27MB 安全下载
康美通 1.0beta
5.86MB 安全下载
下载豌豆荚客户端（更多历史版本）下载
康美通 历史版本年份合集

再去用正则：

```
安全下载
```

替换，得到我们要的：

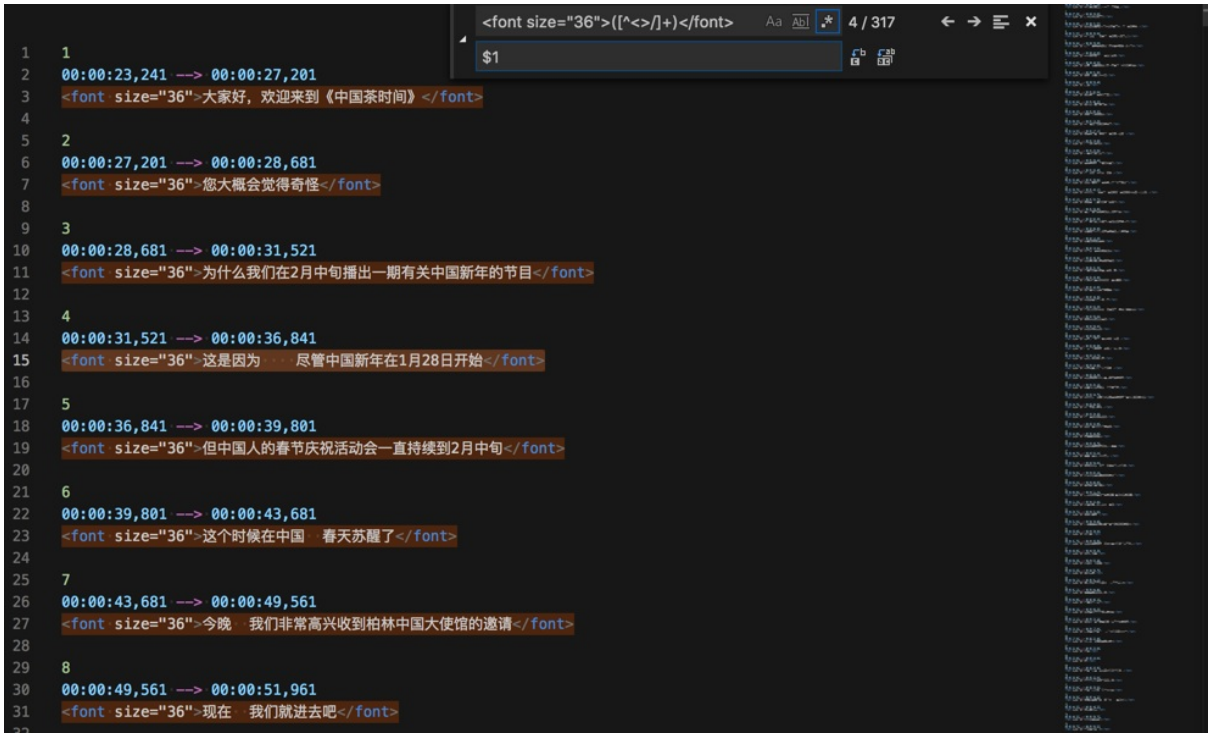
```
相关历史版本:
康美通 4.3.1 23.34MB
康美通 4.3.0 23.34MB
康美通 4.2.3 19.47MB
康美通 4.2.2 19.29MB
康美通 4.2.1 19.29MB
康美通 4.2.0 19.18MB
康美通 4.1.1 19.47MB
康美通 4.1.0 19.48MB
康美通 4.0 14.37MB
康美通 3.2 14.22MB
康美通 3.1 23.2MB
康美通 3.0 23.1MB
康美通 2.0.9 7.93MB
康美通 2.0.8 7.3MB
康美通 2.0.7 7.29MB
康美通 2.0.6 7.29MB
康美通 1.0.1 6.27MB
康美通 1.0beta 5.86MB
```

去掉srt字幕中font size

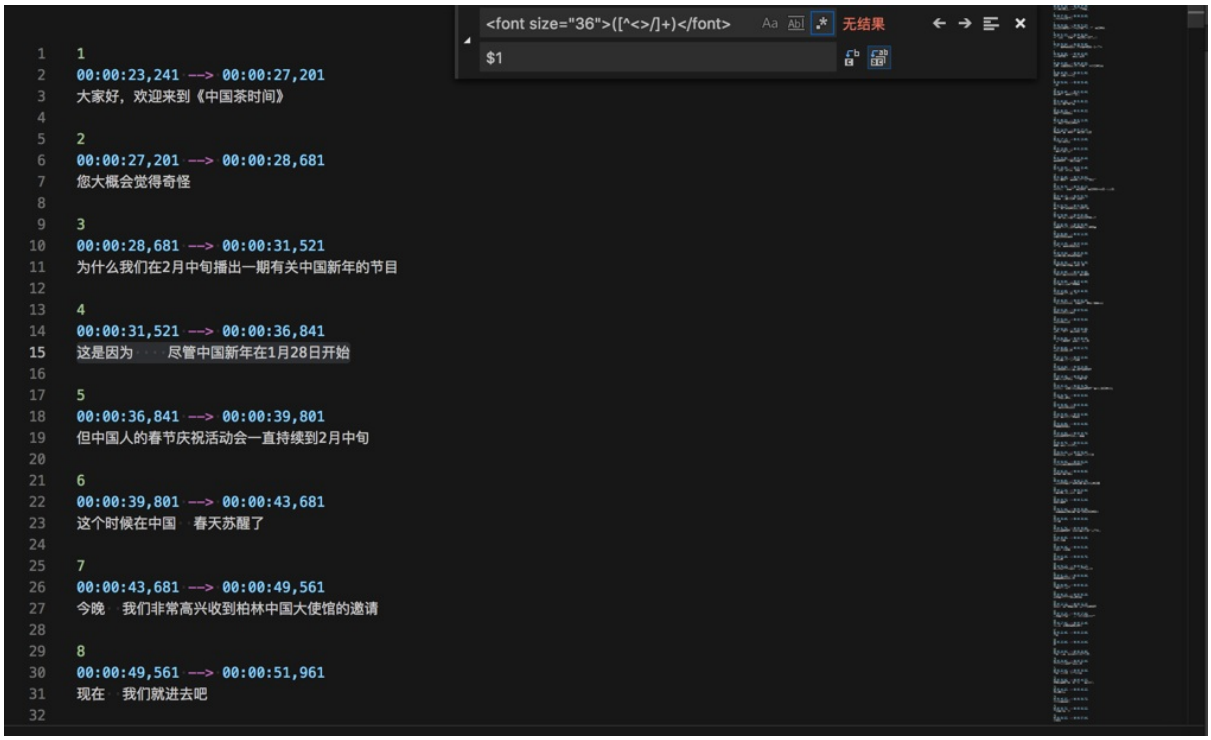
正则：

```
<font size="36">([^</]+)</font>
$1
```

从：



替换成：

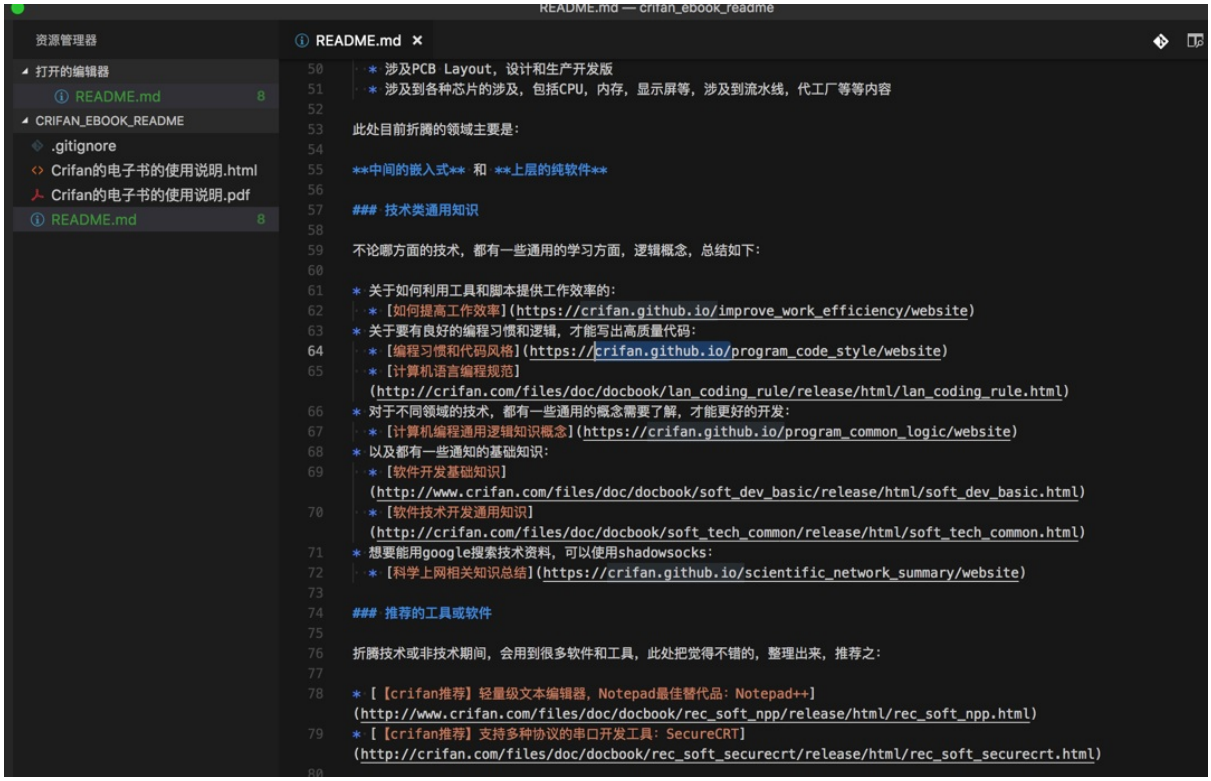


crifan电子书中链接替换

对于我的电子书的说明：

https://github.com/crifan/crifan_ebook_readme

的markdown源码：



想要把其中的地址：

`https://crifan.github.io/xxx/website`

替换为：

`https://book.crifan.com/books/xxx/website/`

比如：

`https://crifan.github.io/program_code_style/website`

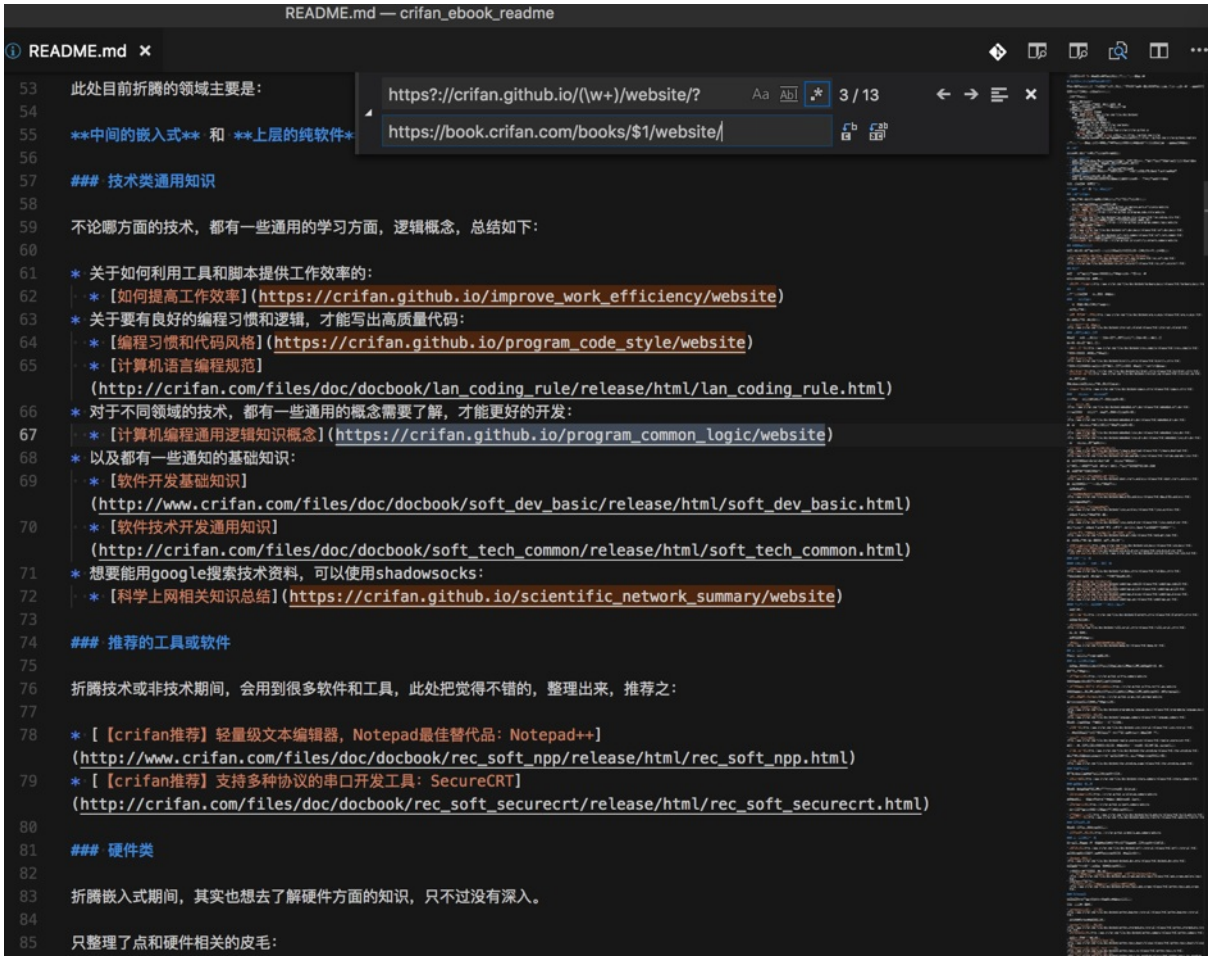
替换成：

`https://book.crifan.com/books/program_code_style/website/`

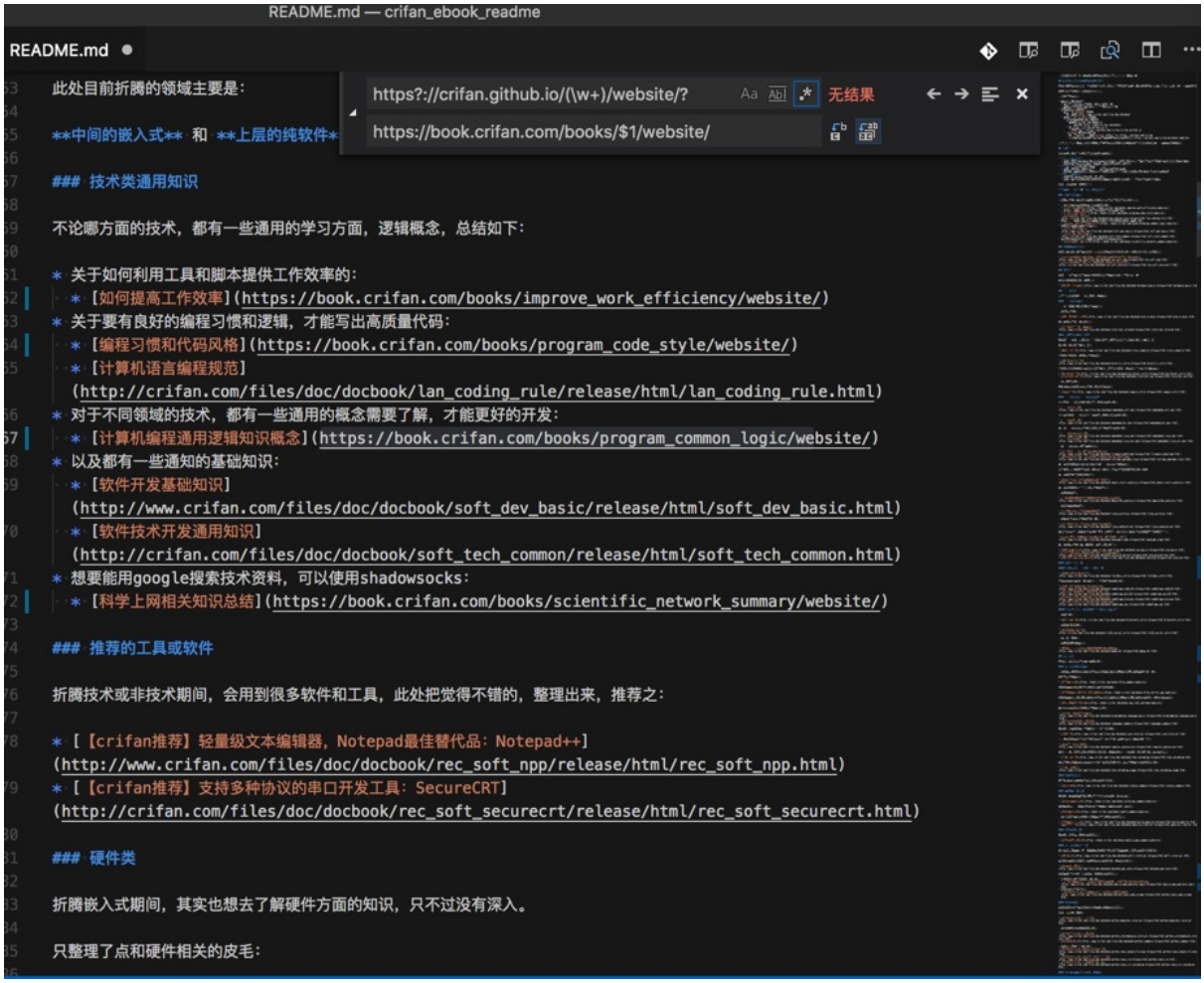
用正则：

```
https?://crifan.github.io/(\w+)/website/?
https://book.crifan.com/books/$1/website/
```

实现从：


README.md — crifan_ebook_readme
README.md
https?://crifan.github.io/(\w+)/website/?
3 / 13
https://book.crifan.com/books/$1/website/
此处目前折腾的领域主要是：
中间的嵌入式 和 **上层的纯软件**
技术类通用知识
不论哪方面的技术，都有一些通用的学习方面，逻辑概念，总结如下：
* 关于如何利用工具和脚本提供工作效率的：
* [如何提高工作效率](https://crifan.github.io/improve_work_efficiency/website)
* 关于要有良好的编程习惯和逻辑，才能写出高质量代码：
* [编程习惯和代码风格](https://crifan.github.io/program_code_style/website)
* [计算机语言编程规范](http://crifan.com/files/doc/docbook/lan_coding_rule/release/html/lan_coding_rule.html)
* 对于不同领域的技术，都有一些通用的概念需要了解，才能更好的开发：
* [计算机编程通用逻辑知识概念](https://crifan.github.io/program_common_logic/website)
* 以及都有一些通知的基础知识：
* [软件开发基础知识](http://www.crifan.com/files/doc/docbook/soft_dev_basic/release/html/soft_dev_basic.html)
* [软件技术开发通用知识](http://crifan.com/files/doc/docbook/soft_tech_common/release/html/soft_tech_common.html)
* 想要能用google搜索技术资料，可以使用shadowsocks：
* [科学上网相关知识总结](https://crifan.github.io/scientific_network_summary/website)
推荐的工具或软件
折腾技术或非技术期间，会用到很多软件和工具，此处把觉得不错的，整理出来，推荐之：
* [【crifan推荐】轻量级文本编辑器，Notepad最佳替代品：Notepad++](http://www.crifan.com/files/doc/docbook/rec_soft_npp/release/html/rec_soft_npp.html)
* [【crifan推荐】支持多种协议的串口开发工具：SecureCRT](http://crifan.com/files/doc/docbook/rec_soft_securecrt/release/html/rec_soft_securecrt.html)
硬件类
折腾嵌入式期间，其实也想去了解硬件方面的知识，只不过没有深入。
只整理了点和硬件相关的皮毛：

替换成：



将Chrome中拷贝出来的cookie处理成代码中要的dict

Chrome中拷贝出来的cookie是：

```
welcomeflash 20050606_107001; zzpaneluin ; zzpanelkey ; pgv_pvi 7640393728; pgv_si s314729
8816; pgv_pvid 3951804270; pgv_info ssid s7487670374; ptisp ctc; ptui_loginuin 2539619267;
 pt2gguin o2539619267; uin o2539619267; skey @nDTkOJm1m; RK Ye5Jmtb0ly; ptcz 3b6806bf7cddc
375bc2d23b04ff9c366b47fac808b9256322ad69a23f2dc580f; p_uin o2539619267; pt4_token aC5vGfNA
A2M3fS7ngcAHdXoiCvqwrAGcEuL54gs63oE_; p_skey kSC7q75Gk93gLlo*mRMg*h2m3iYUuubjQqVBIgEMi*o_;
 Loading Yes
```

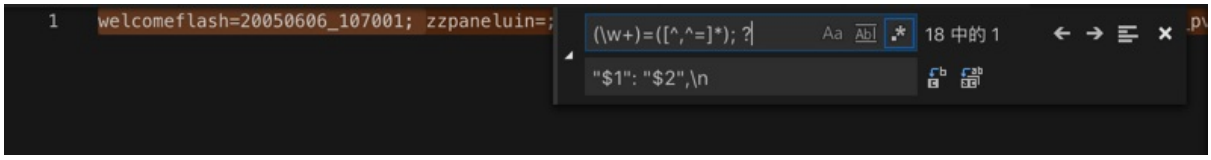
最后加上分号：

```
welcomeflash 20050606_107001; zzpaneluin ; zzpanelkey ; pgv_pvi 7640393728; pgv_si s314729
8816; pgv_pvid 3951804270; pgv_info ssid s7487670374; ptisp ctc; ptui_loginuin 2539619267;
 pt2gguin o2539619267; uin o2539619267; skey @nDTkOJm1m; RK Ye5Jmtb0ly; ptcz 3b6806bf7cddc
375bc2d23b04ff9c366b47fac808b9256322ad69a23f2dc580f; p_uin o2539619267; pt4_token aC5vGfNA
A2M3fS7ngcAHdXoiCvqwrAGcEuL54gs63oE_; p_skey kSC7q75Gk93gLlo*mRMg*h2m3iYUuubjQqVBIgEMi*o_;
 Loading Yes;
```

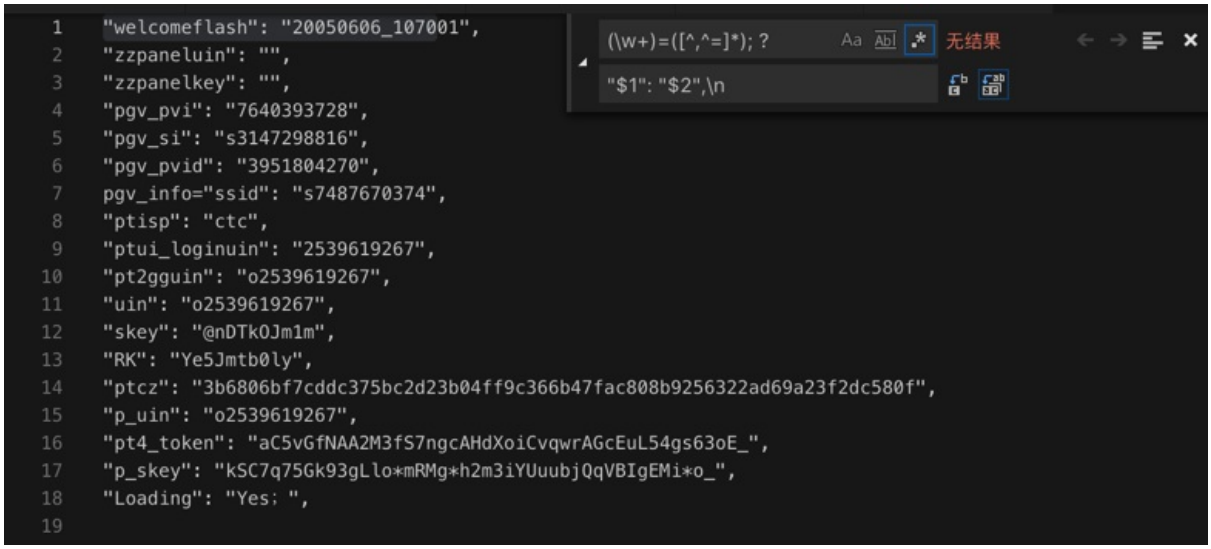
再去用正则：

```
(\w+)=([^,^=]*); ?
"$1": "$2",\n
```

替换：



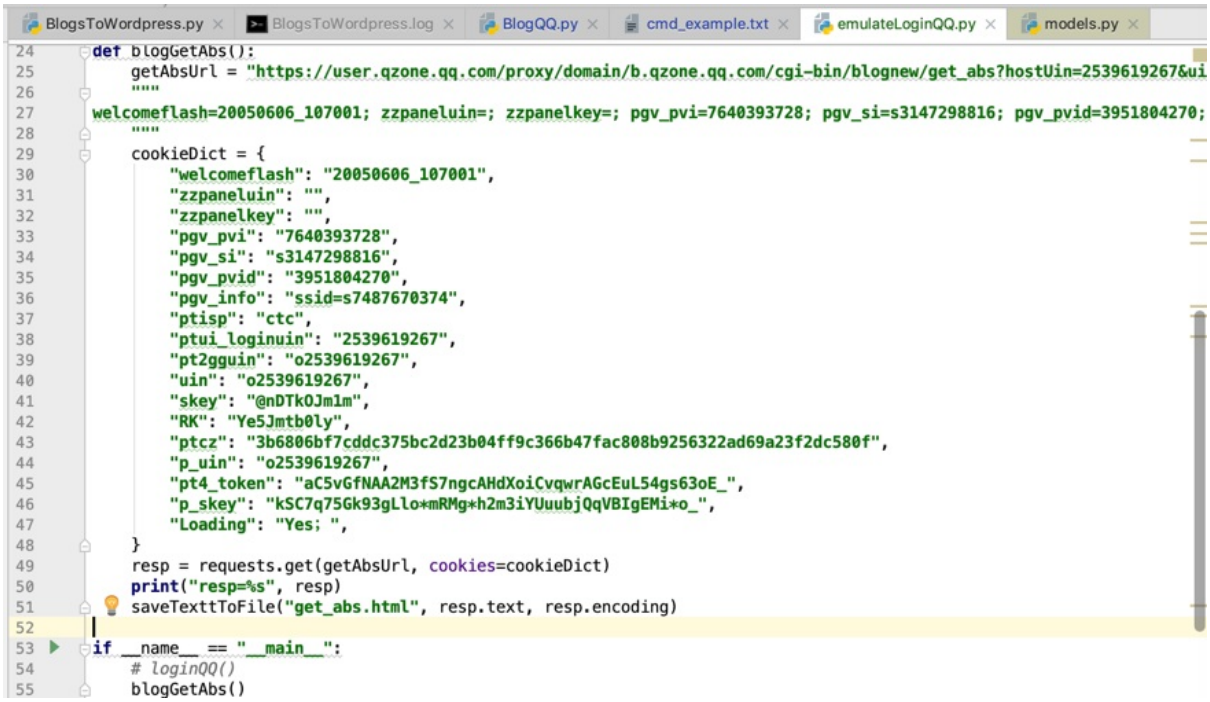
成自己要的dict的内容：



中间有个特殊的，自己手动改一下即可：

```
"welcomeflash": "20050606_107001",
"zzpaneluin": "",
"zzpanelkey": "",
"pgv_pvi": "7640393728",
"pgv_si": "s3147298816",
"pgv_pvid": "3951804270",
"pgv_info": "ssid=s7487670374",
"ptisp": "ctc",
"ptui_loginuin": "2539619267",
"pt2gguin": "o2539619267",
"uin": "o2539619267",
"skey": "@nDTkOJm1m",
"RK": "Ye5Jmtb0ly",
"ptcz": "3b6806bf7cddc375bc2d23b04ff9c366b47fac808b9256322ad69a23f2dc580f",
"p_uin": "o2539619267",
"pt4_token": "aC5vGfNAA2M3fS7ngcAHdXoiCvqwrAGcEuL54gs63oE_",
"p_skey": "kSC7q75Gk93gLlo*mRMg*h2m3iYUuubjQqVBIgEMi*o_",
"Loading": "Yes;",
```

粘贴到代码中即可使用了：



处理得到城市名称

除了：

【整理】中国常见的城市的名字

以及：

对于：

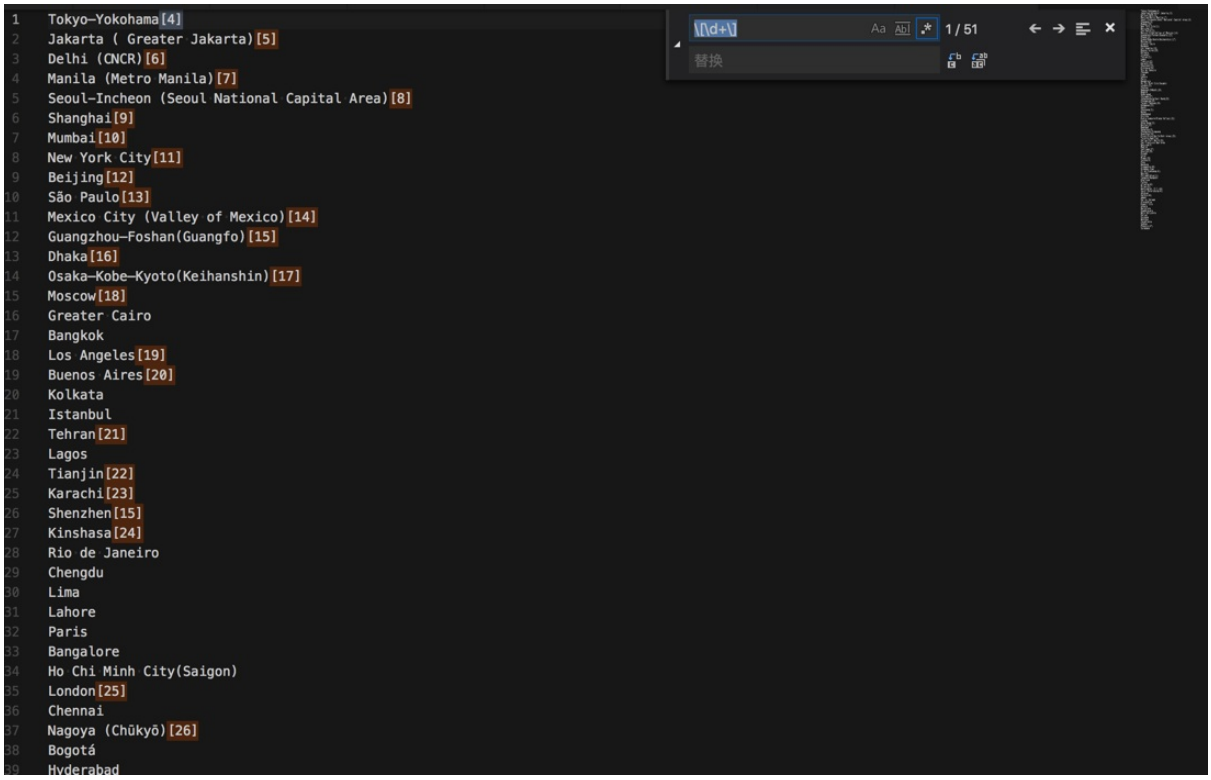
https://en.wikipedia.org/wiki/List_of_urban_areas_by_population

中的城市名，用正则：

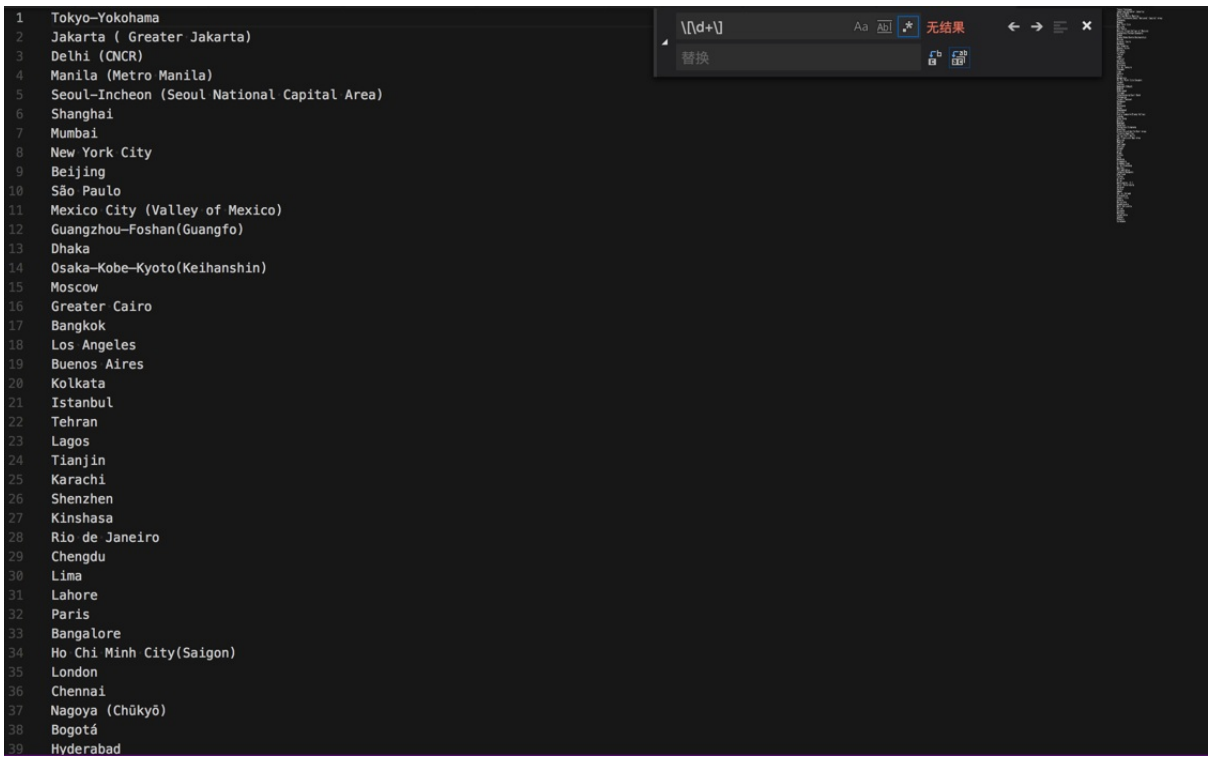
```
\[\d+\]
```

去除掉`[数字]`

从：



替换成：

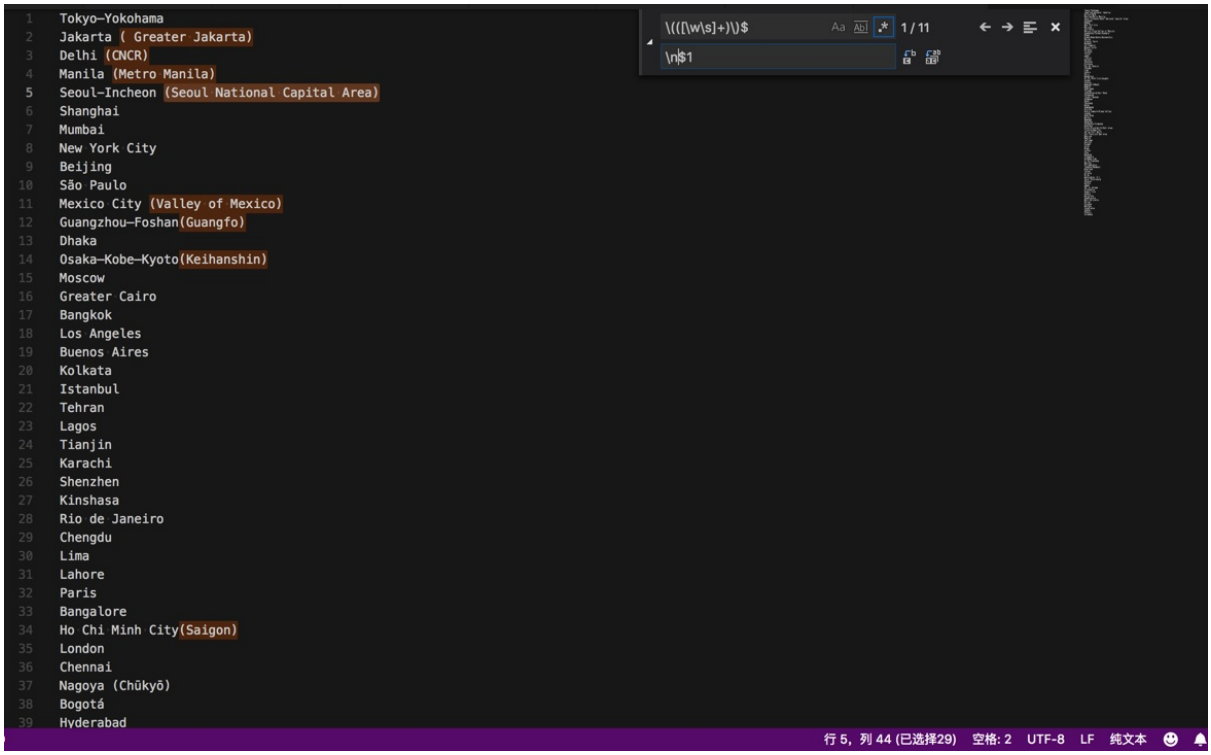


以及继续用：

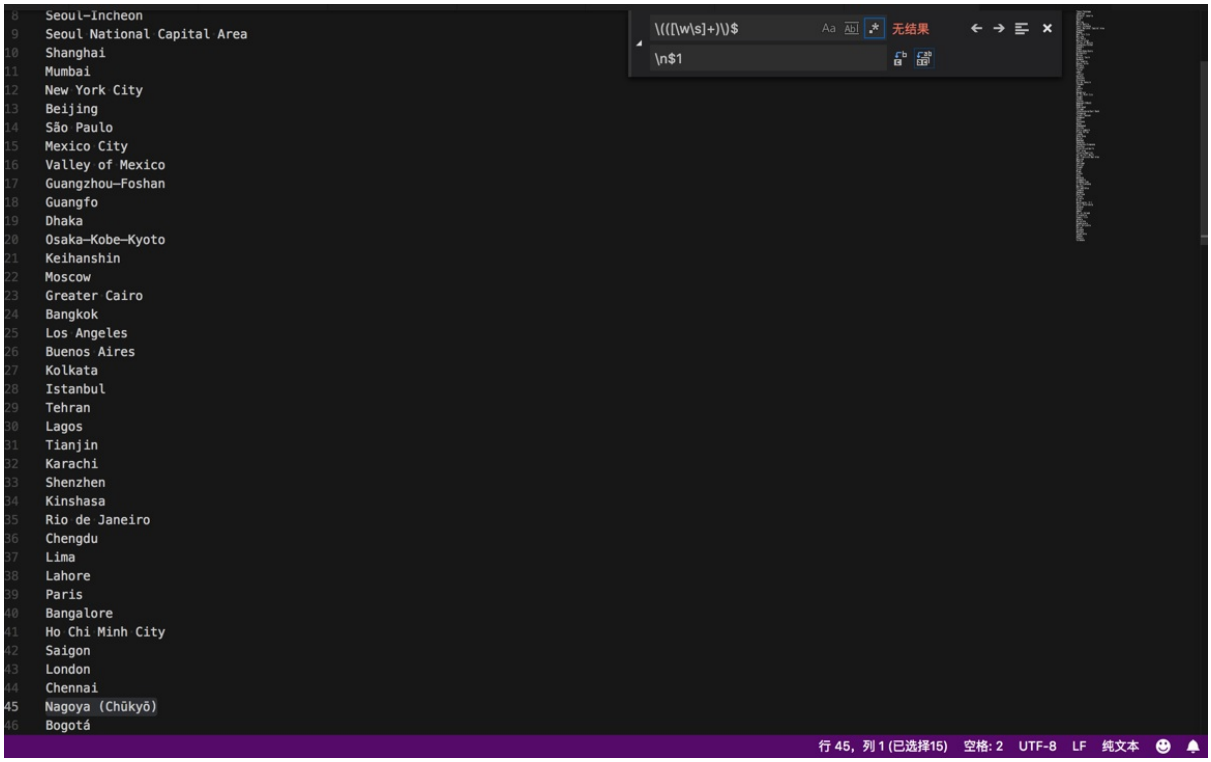
```
\(([\w\s]+)\)$
$1
```

把 `(xxx)` 中的 `xxx` 放到下一行

从：



替换成：



当然，也注意到了，没有匹配到：

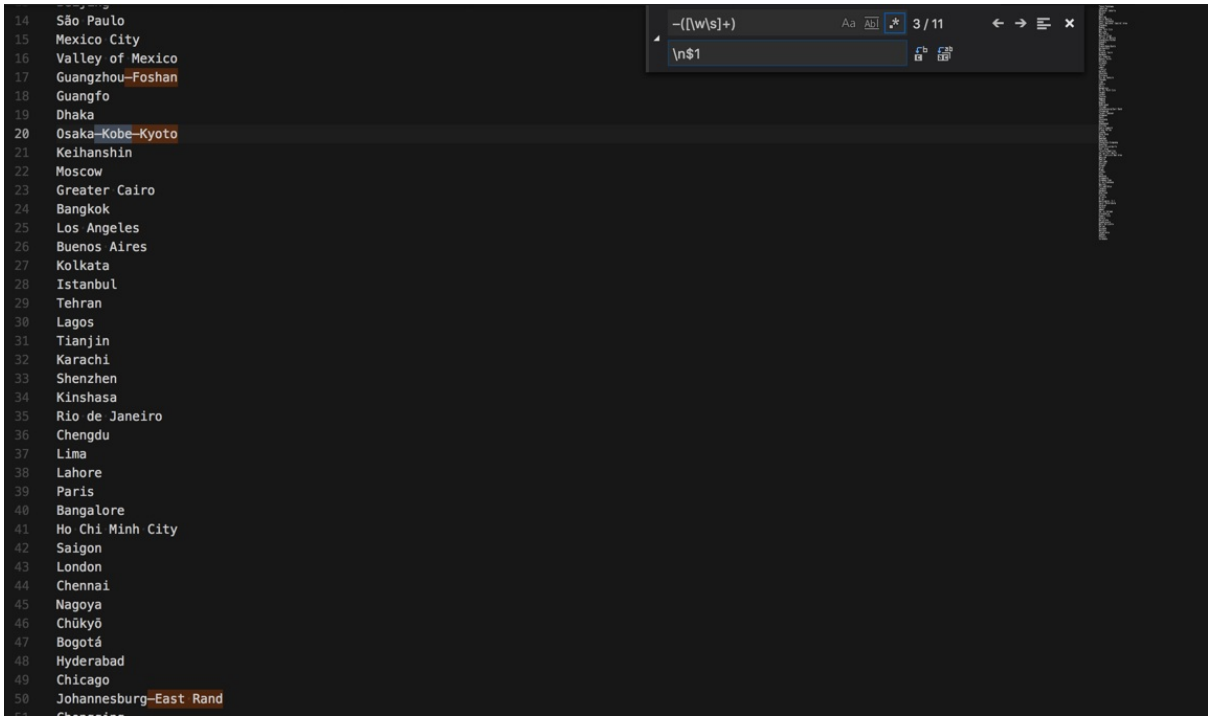
`Nagoya (Chūkyō)`

是因为里面有unicode的字符，由于数量不多，手动处理即可。

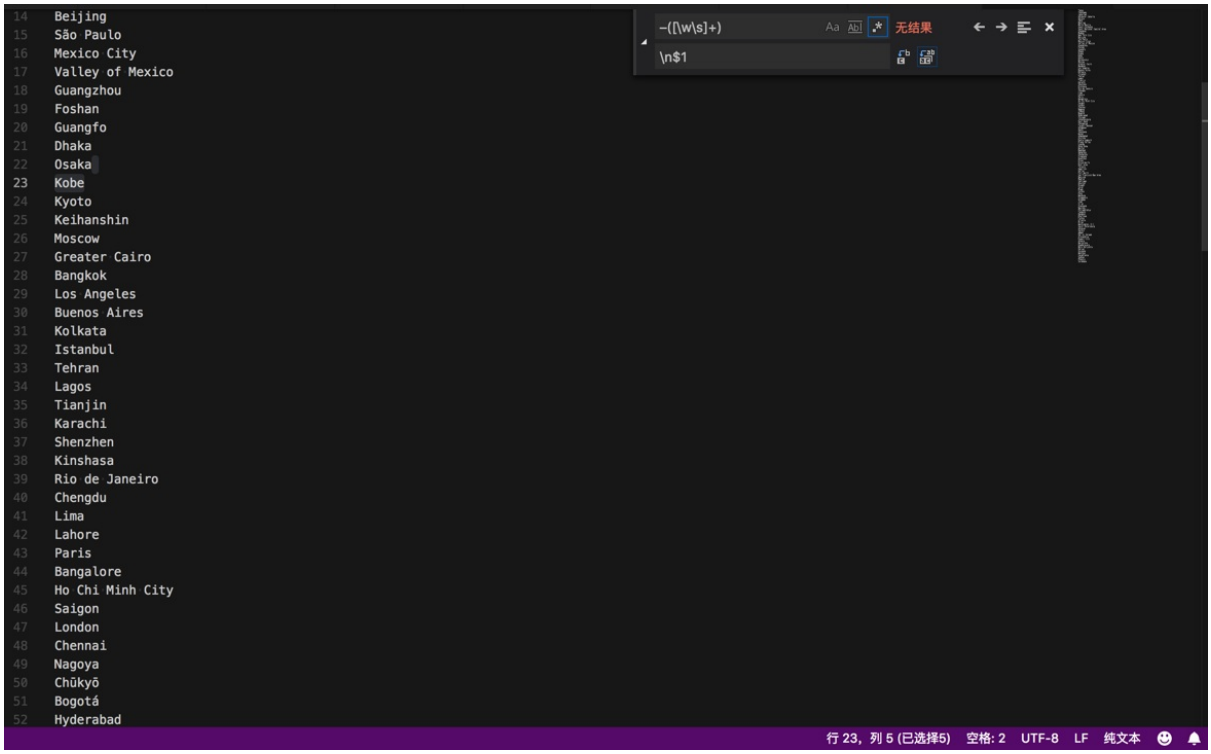
再去用：

```
-([\w\s]+)
\n$1
```

把 `xxx-yyy` 中的 `yyy` 放到下一行



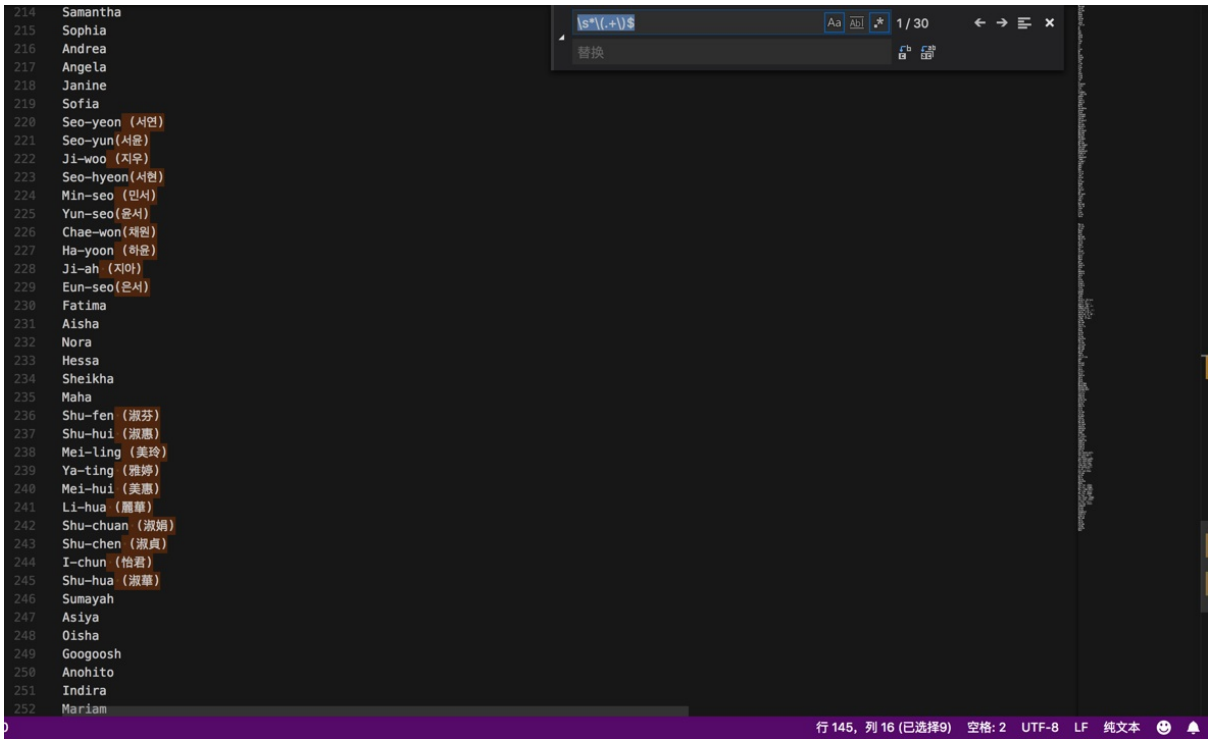
替换成：



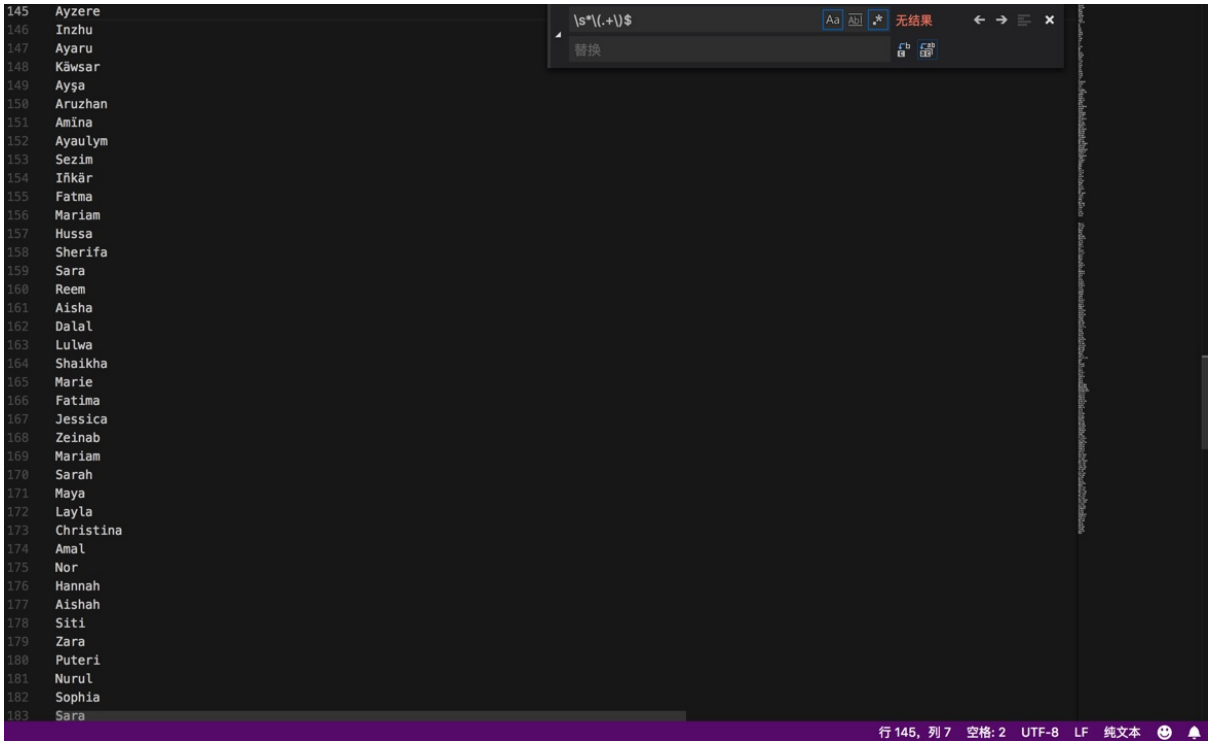
以及，用：

```
\s*\(.+\)$
```

把 `xxx（yyy）` 中的 `空格(yyy)` 去掉：



替换成：



再继续，用：

```
^\s
\n
```

可以找到：

有哪些单词在行首有多余的空格（后续可以再去删除掉）：

![vscode_remove_start_empty.png)

提取mp3文件名和mp3链接地址

正则：

```
^[^\r\n]+href "(\w+\](../../assets/img/vscode_remove_start_empty.png)

## 提取mp3文件名和mp3链接地址

正则：

```bash
^[^\r\n]+href="(\w+\.mp3)"[^\r\n]+$
$1
```

把：

```
<img src="/icons/sound2.gif" alt="[SND]"> <a href="e10d3a.mp3">e10d3a.mp3</a> 2015-09-23 0
```
```

```
9:10 9.3M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="e10d3b.mp3">e10d3b.mp3</a> 2015-09-23 0
9:10 7.0M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="e10d5a.mp3">e10d5a.mp3</a> 2015-09-23 0
9:11 54M
```

替换为：

```
e10d3a.mp3
e10d3b.mp3
e10d5a.mp3
```

再进一步：

用正则：

```
^[^\r\n]+href="(\w+\.mp3)"[^\r\n]+$
http://media.talkbank.org/CHILDES/Biling/Singapore/$1
```

从：

```
<img src="/icons/sound2.gif" alt="[SND]"> <a href="e10d3a.mp3">e10d3a.mp3</a> 2015-09-23 0
9:10 9.3M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="e10d3b.mp3">e10d3b.mp3</a> 2015-09-23 0
9:10 7.0M
<img src="/icons/sound2.gif" alt="[SND]"> <a href="e10d5a.mp3">e10d5a.mp3</a> 2015-09-23 0
9:11 54M
```

替换和提取出：

```
http://media.talkbank.org/CHILDES/Biling/Singapore/e10d3a.mp3
http://media.talkbank.org/CHILDES/Biling/Singapore/e10d3b.mp3
http://media.talkbank.org/CHILDES/Biling/Singapore/e10d5a.mp3
```

详见：

【已解决】VSCode中如何使用正则表达式去替换且被替换中使用分组group

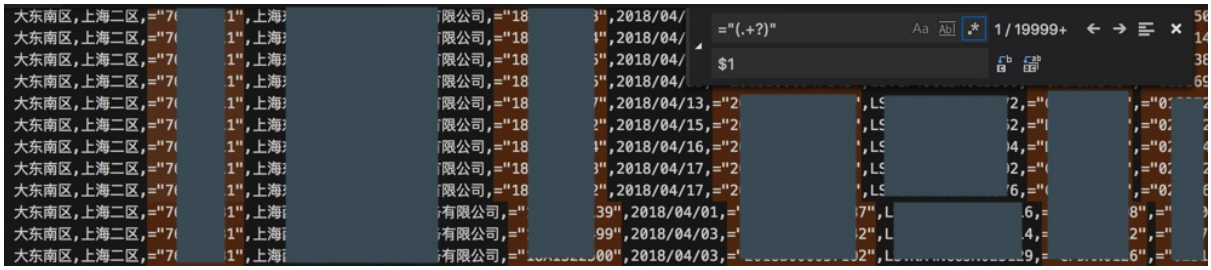
去除掉csv中多余的 `="xxx"`

客户给的一个数据文件csv格式的，但是内部内容中发现有多余的 `="xxx"` ，应该改为 `xxx` 才对。

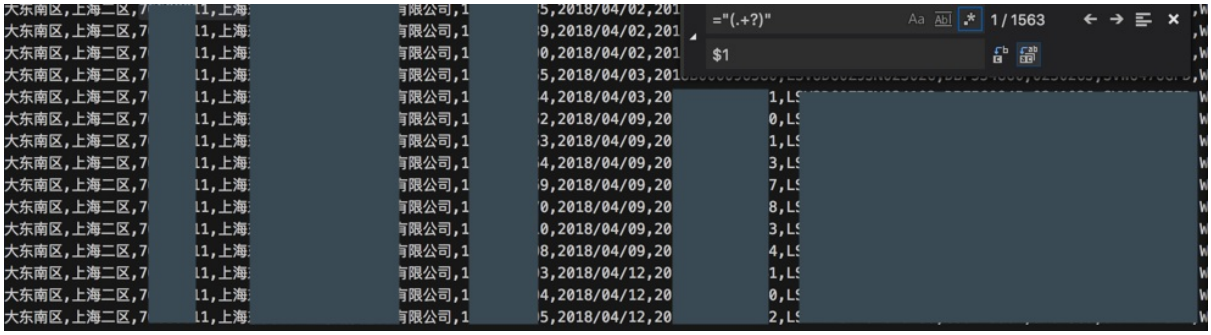
所以用VSCode去替换，用正则：

```
="(.+?)"
$1
```

实现了，把 `="7xxx1"`：



替换成 `7xxx1`：



即可。

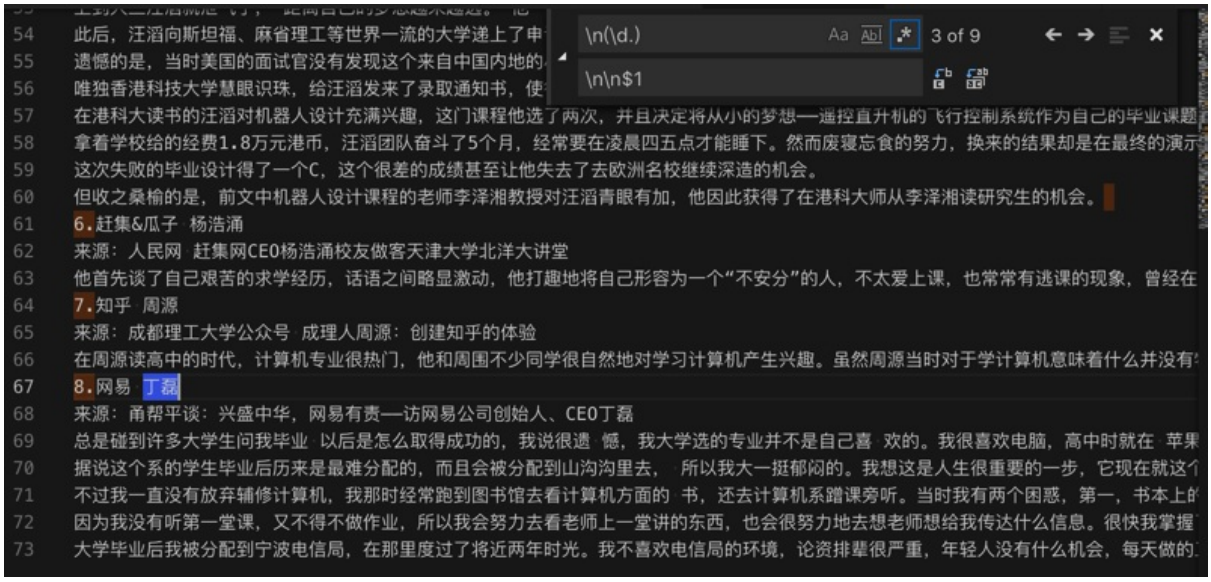
给文章段落增加换行

用正则：

```
\n(\d.)
\n\n$1
```

把：

```
...互。
1.字节跳动
...说过。
2.陆奇给年轻人的话
...
8.网易 丁磊
...
```

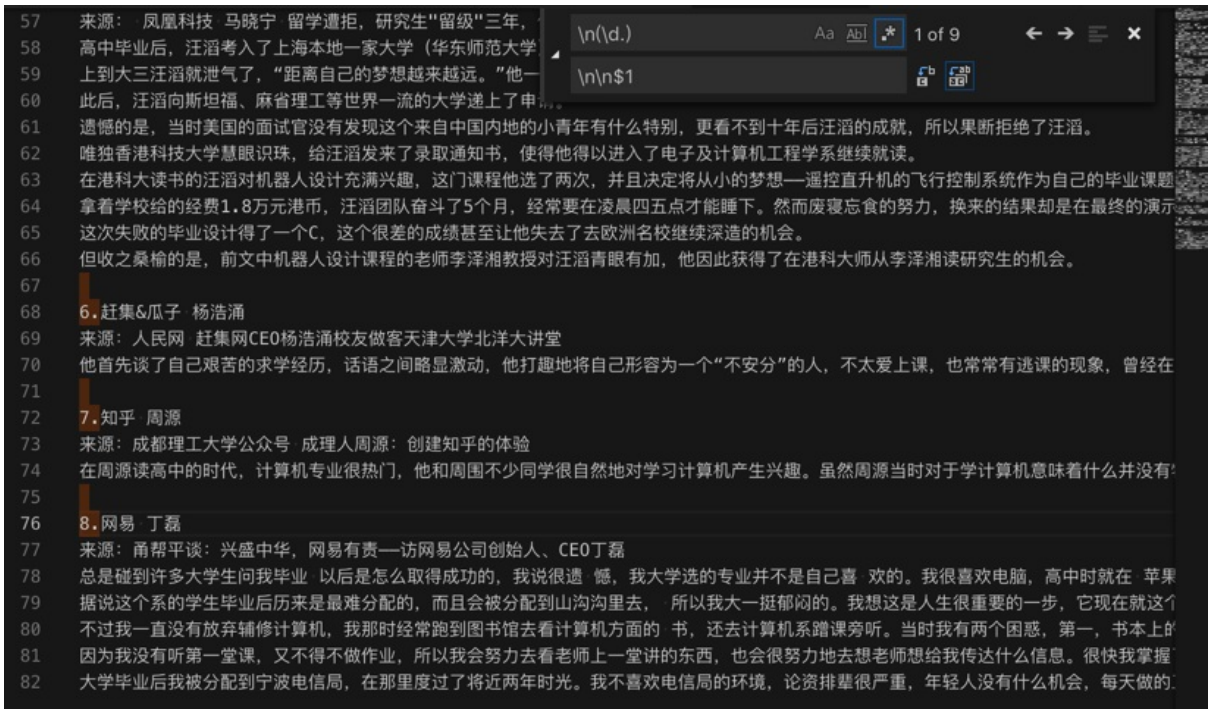


变成：

...互。
1.字节跳动
...说过。

2.陆奇给年轻人的话
...

8.网易 丁磊
...



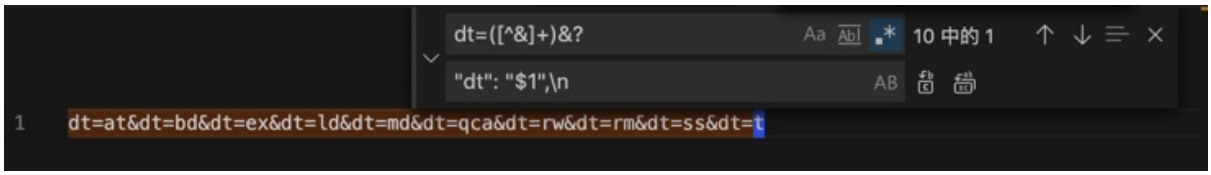
把url中查询参数换成代码中字典参数

用正则：

```
dt ([^&]+)&?
"dt": "$1",\n
```

把：

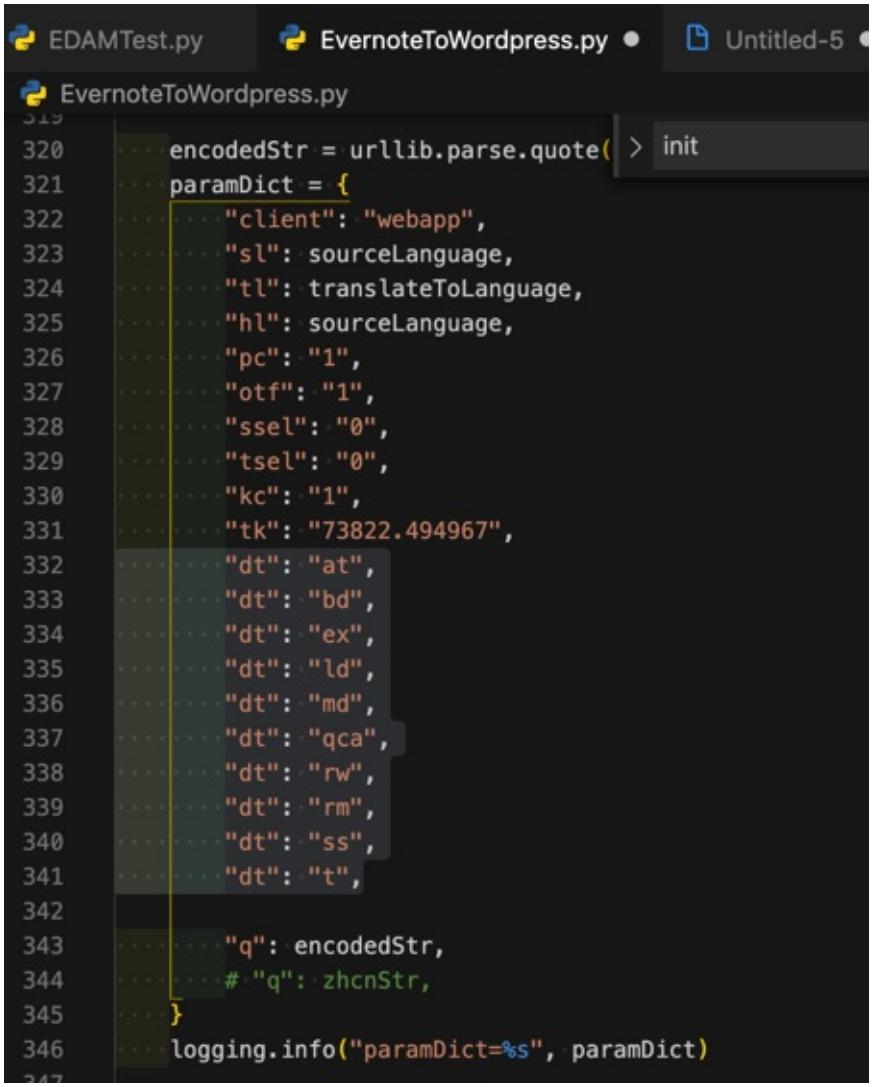
```
dt at dt bd dt ex dt ld dt md dt qca dt rw dt rm dt ss dt t
```



替换成：

```
"dt": "at",
"dt": "bd",
"dt": "ex",
"dt": "ld",
"dt": "md",
"dt": "qca",
"dt": "rw",
"dt": "rm",
"dt": "ss",
"dt": "t",
```

用于后续放到代码中使用：



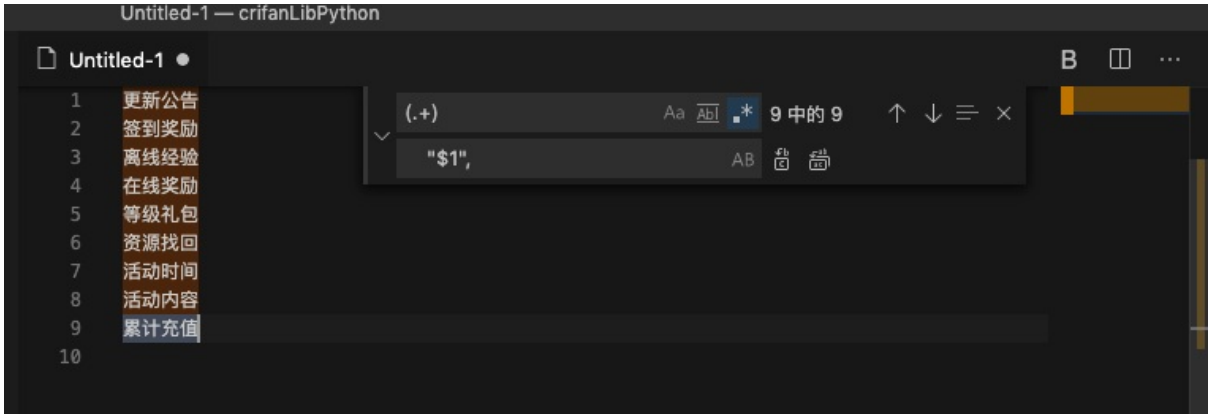
把每个词都加上引号，用于放代码中用

用正则：

```
(.+)
    "$1",
```

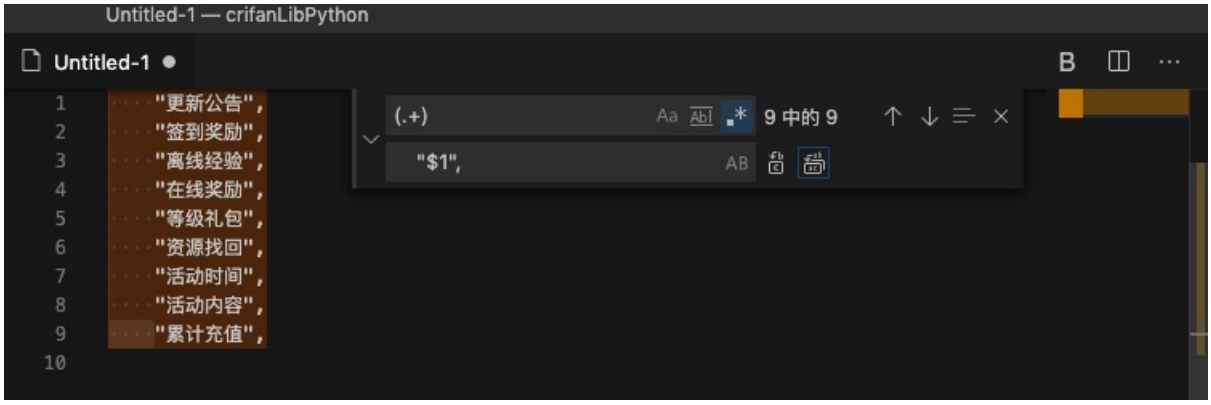
从输入：

```
更新公告
签到奖励
离线经验
在线奖励
等级礼包
资源找回
活动时间
活动内容
累计充值
```

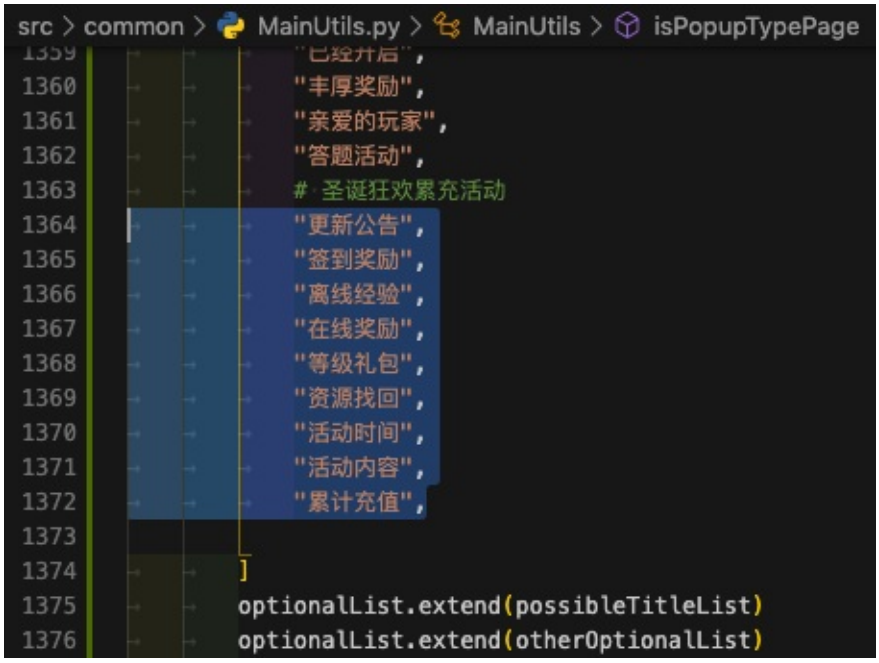


变成：

```
    "更新公告",
    "签到奖励",
    "离线经验",
    "在线奖励",
    "等级礼包",
    "资源找回",
    "活动时间",
    "活动内容",
    "累计充值",
```



用于拷贝到代码中使用：



提取104协议示例数据，并格式化成java代码中字符串数组

从：

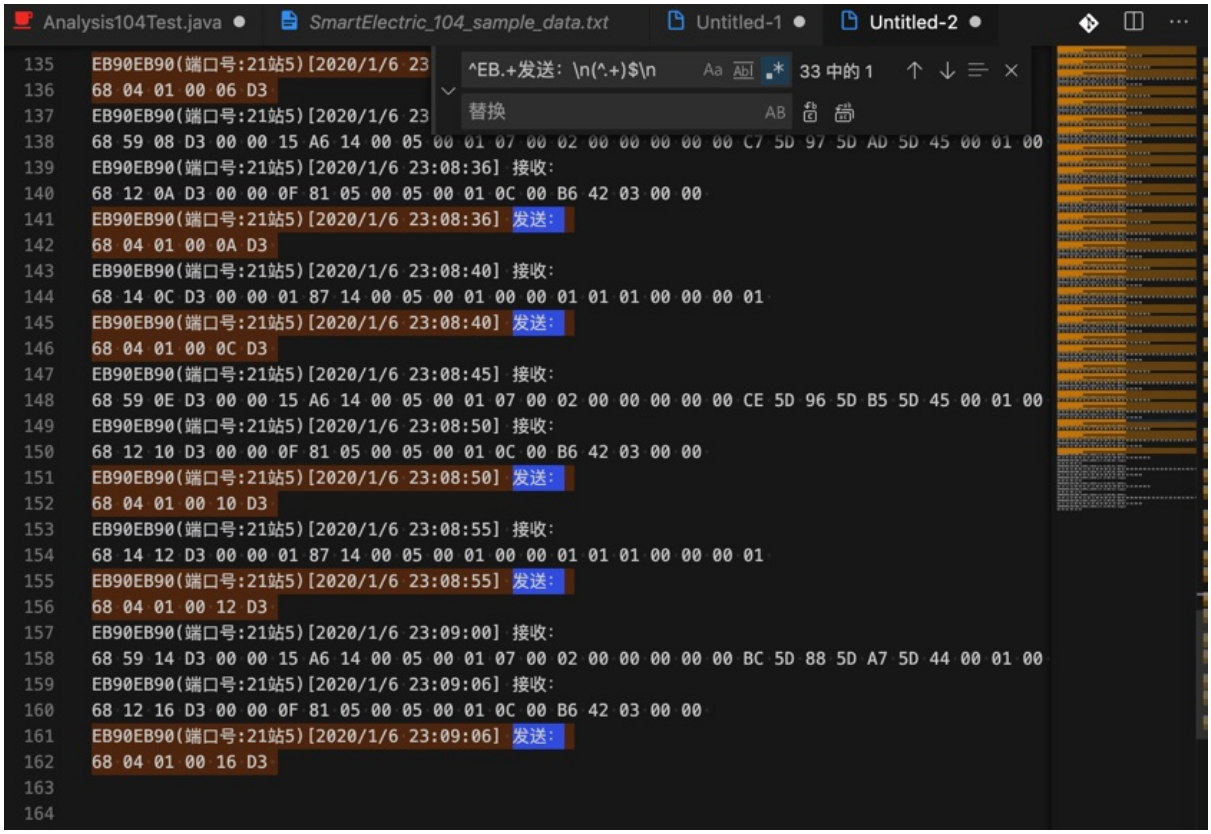
```
EB90EB90(端口号:21站0)[2020/1/6 18:18:58] 发送:
68 04 07 00 00 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:07] 接收:
68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:07] 发送:
68 04 01 00 E6 B7
EB90EB90(端口号:21站5)[2020/1/6 18:19:12] 接收:
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 18:19:12] 发送:
68 04 01 00 E8 B7
EB90EB90(端口号:21站5)[2020/1/6 18:19:17] 接收:
68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5C DF 5C 45 00 01
00 FE FF 7C 00 94 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 86 13 00 00 02 00 00 00 00 00 FE FF 00 00 00 00 87 00 00 00 00 00 02 00 00 00 00
00
...
EB90EB90(端口号:21站5)[2020/1/6 23:08:55] 发送:
68 04 01 00 12 D3
EB90EB90(端口号:21站5)[2020/1/6 23:09:00] 接收:
68 59 14 D3 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 BC 5D 88 5D A7 5D 44 00 01
00 FE FF 7C 00 84 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 88 13 00 00 02 00 00 00 00 00 FE FF 00 00 00 00 B6 00 00 00 00 00 02 00 00 00 00
00
EB90EB90(端口号:21站5)[2020/1/6 23:09:06] 接收:
68 12 16 D3 00 00 0F 81 05 00 05 00 01 0C 00 B6 42 03 00 00
EB90EB90(端口号:21站5)[2020/1/6 23:09:06] 发送:
68 04 01 00 16 D3
```

提取出：接收：的下一行的一连串数字

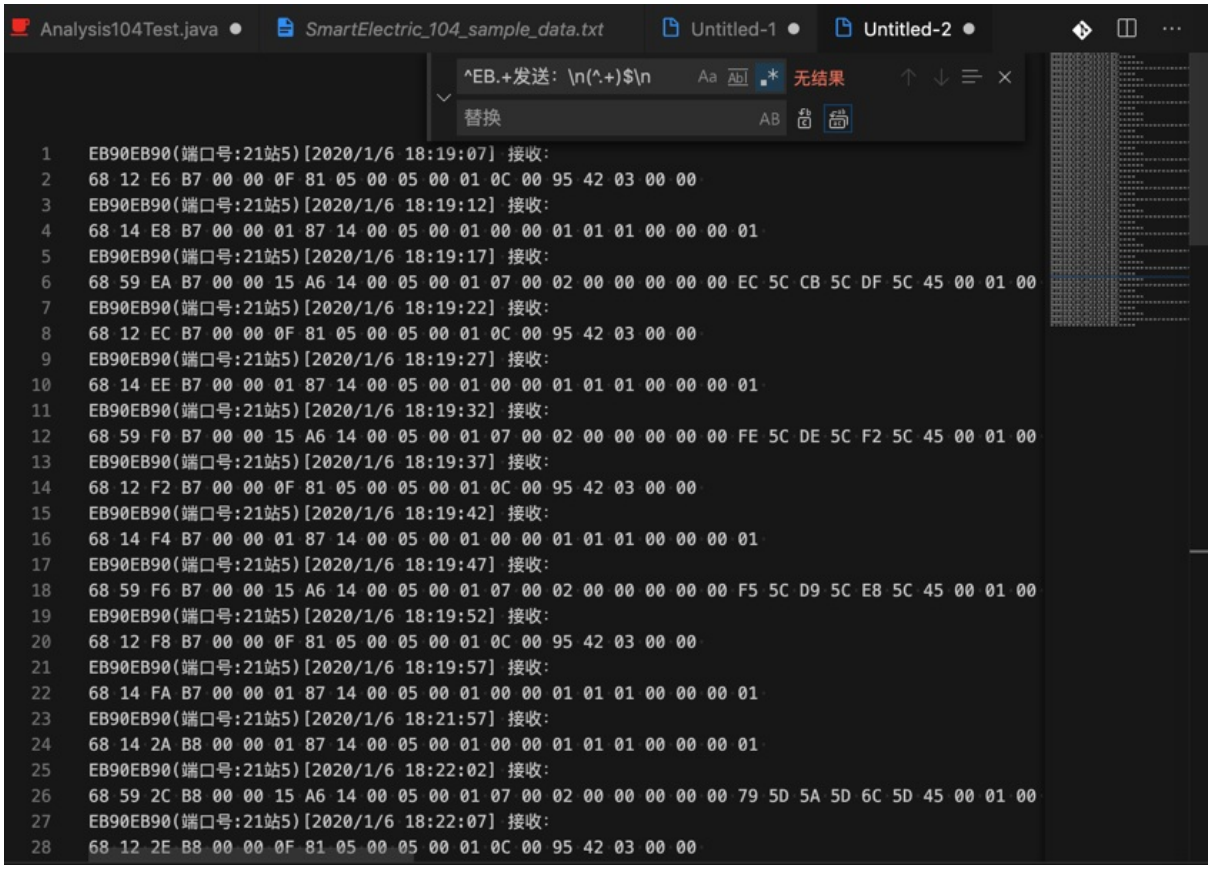
（1）先 去除 发送的部分

正则：

```
^EB.+发送：\n(^.+)$\n
```



替换成：



（2）再去把接收部分中数字提取出来

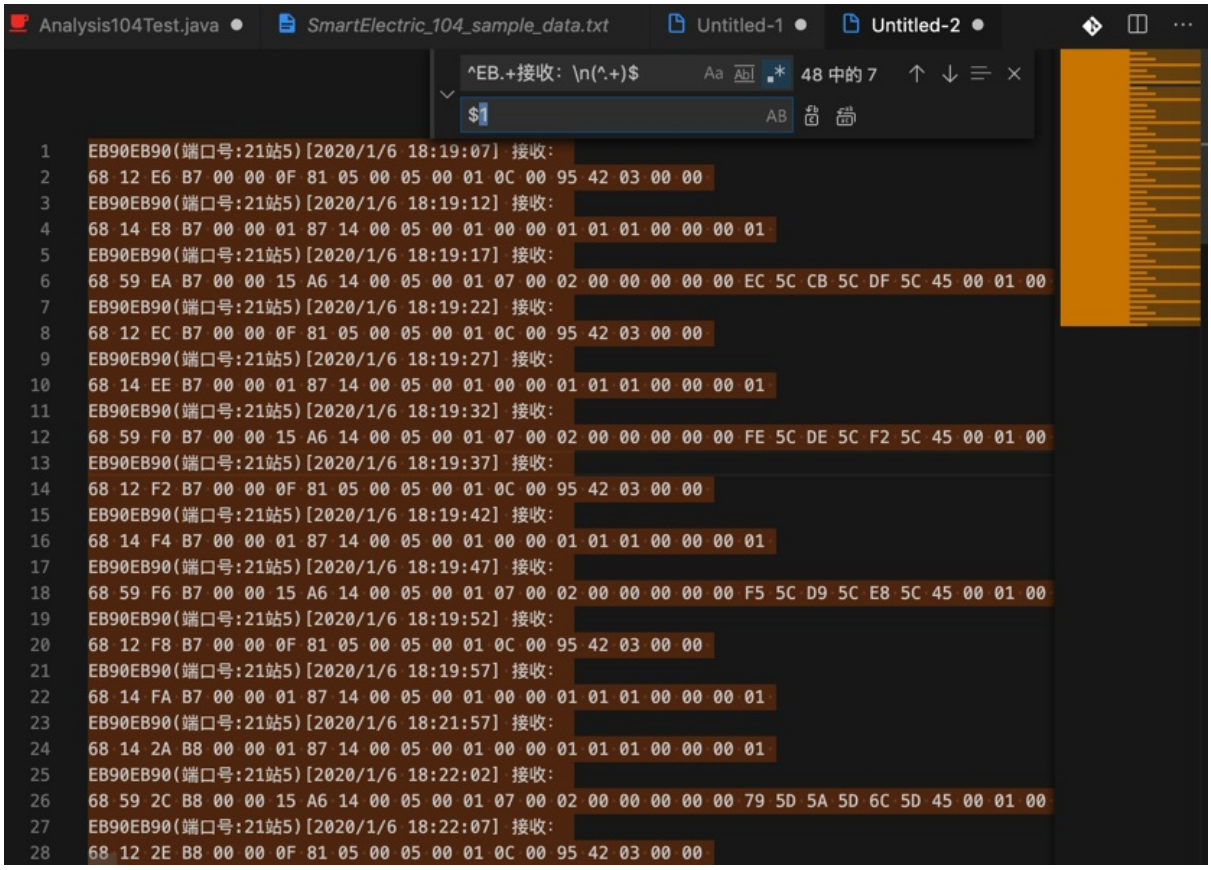
从：

```
EB90EB90(端口号:21站5)[2020/1/6 18:19:07] 接收:
68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:12] 接收:
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
...
EB90EB90(端口号:21站5)[2020/1/6 23:08:55] 接收:
68 14 12 D3 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 23:09:00] 接收:
68 59 14 D3 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 BC 5D 88 5D A7 5D 44 00 01
00 FE FF 7C 00 84 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 88 13 00 00 02 00 00 00 00 00 FE FF 00 00 00 00 B6 00 00 00 00 00 02 00 00 00 00
00
EB90EB90(端口号:21站5)[2020/1/6 23:09:06] 接收:
68 12 16 D3 00 00 0F 81 05 00 05 00 01 0C 00 B6 42 03 00 00
```

用正则：

```
^EB.+接收: \n(^.+)$
$1
```

把：


Analysis104Test.java
SmartElectric_104_sample_data.txt
Untitled-1
Untitled-2
^EB.+接收：\n(^.+)$
48 中的 7
$1
EB90EB90(端口号:21站5)[2020/1/6 18:19:07] 接收:
68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:12] 接收:
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 18:19:17] 接收:
68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5C DF 5C 45 00 01 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:22] 接收:
68 12 EC B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:27] 接收:
68 14 EE B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 18:19:32] 接收:
68 59 F0 B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 FE 5C DE 5C F2 5C 45 00 01 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:37] 接收:
68 12 F2 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:42] 接收:
68 14 F4 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 18:19:47] 接收:
68 59 F6 B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 F5 5C D9 5C E8 5C 45 00 01 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:52] 接收:
68 12 F8 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
EB90EB90(端口号:21站5)[2020/1/6 18:19:57] 接收:
68 14 FA B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 18:21:57] 接收:
68 14 2A B8 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 18:22:02] 接收:
68 59 2C B8 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 79 5D 5A 5D 6C 5D 45 00 01 00
EB90EB90(端口号:21站5)[2020/1/6 18:22:07] 接收:
68 12 2E B8 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00

替换成：


Analysis104Test.java
SmartElectric_104_sample_data.txt
Untitled-1
Untitled-2
^EB.+接收：\n(^.+)$
无结果
$1
68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5C DF 5C 45 00 01 00
68 12 EC B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 EE B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 F0 B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 FE 5C DE 5C F2 5C 45 00 01 00
68 12 F2 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 F4 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 F6 B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 F5 5C D9 5C E8 5C 45 00 01 00
68 12 F8 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 FA B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 14 2A B8 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 2C B8 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 79 5D 5A 5D 6C 5D 45 00 01 00
68 12 2E B8 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 30 B8 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 32 B8 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 7E 5D 60 5D 6F 5D 44 00 01 00
68 12 34 B8 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 36 B8 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 38 B8 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 7E 5D 67 5D 71 5D 45 00 01 00
68 14 72 B8 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 74 B8 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 70 5D 4E 5D 4F 5D 44 00 01 00
68 12 76 B8 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 78 B8 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 7A B8 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 6B 5D 47 5D 61 5D 44 00 01 00
68 12 7C B8 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 59 EA D2 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 B9 5D 7E 5D 9D 5D 45 00 01 00
68 12 EC D2 00 00 0F 81 05 00 05 00 01 0C 00 B5 42 03 00 00
68 14 EE D2 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01

得到每一行的数字：

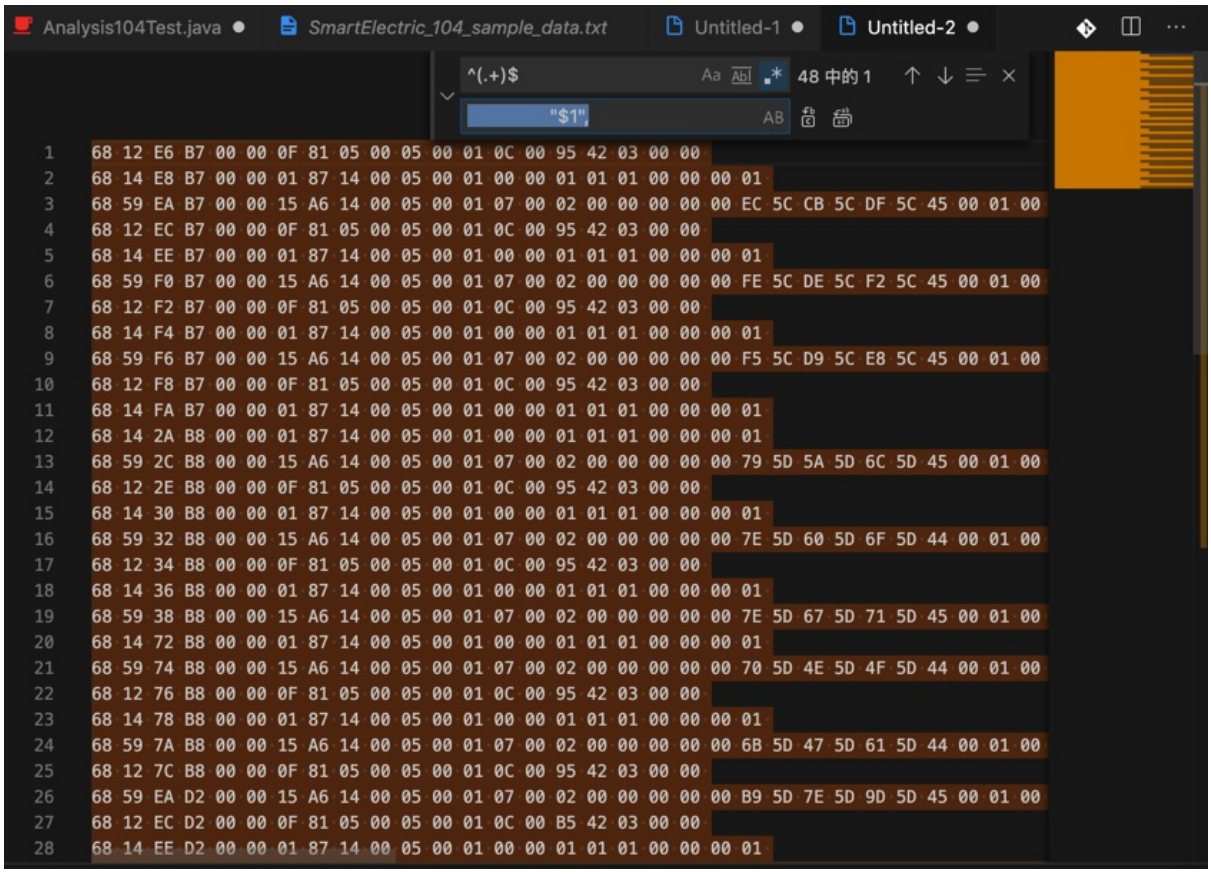
```
68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5C DF 5C 45 00 01
00 FE FF 7C 00 94 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 86 13 00 00 02 00 00 00 00 00 FE FF 00 00 00 00 87 00 00 00 00 00 02 00 00 00 00
00
...
68 12 16 D3 00 00 0F 81 05 00 05 00 01 0C 00 B6 42 03 00 00
```

（3）再去变成java字符串数组，即给每一行加上前后双引号

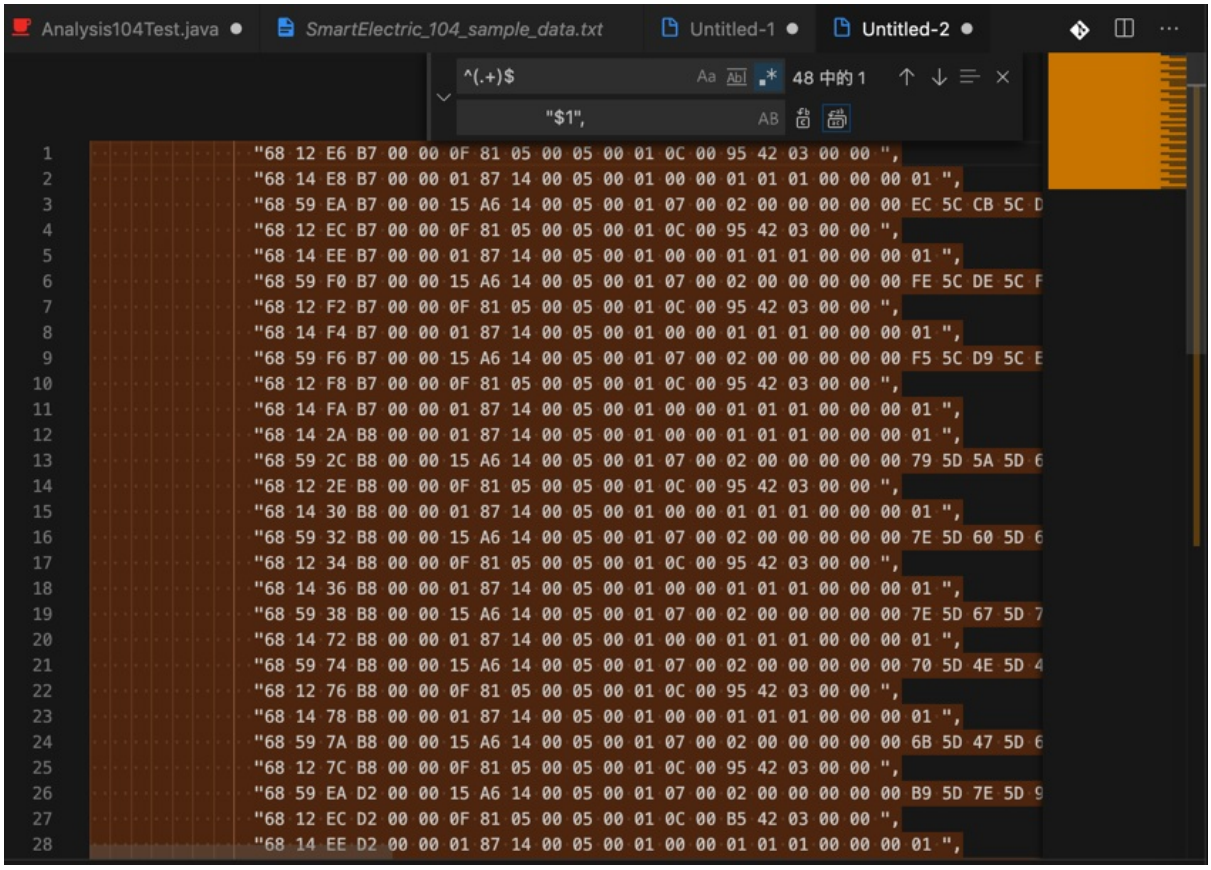
用正则：

```
^(.+)$
                "$1",
```

把：

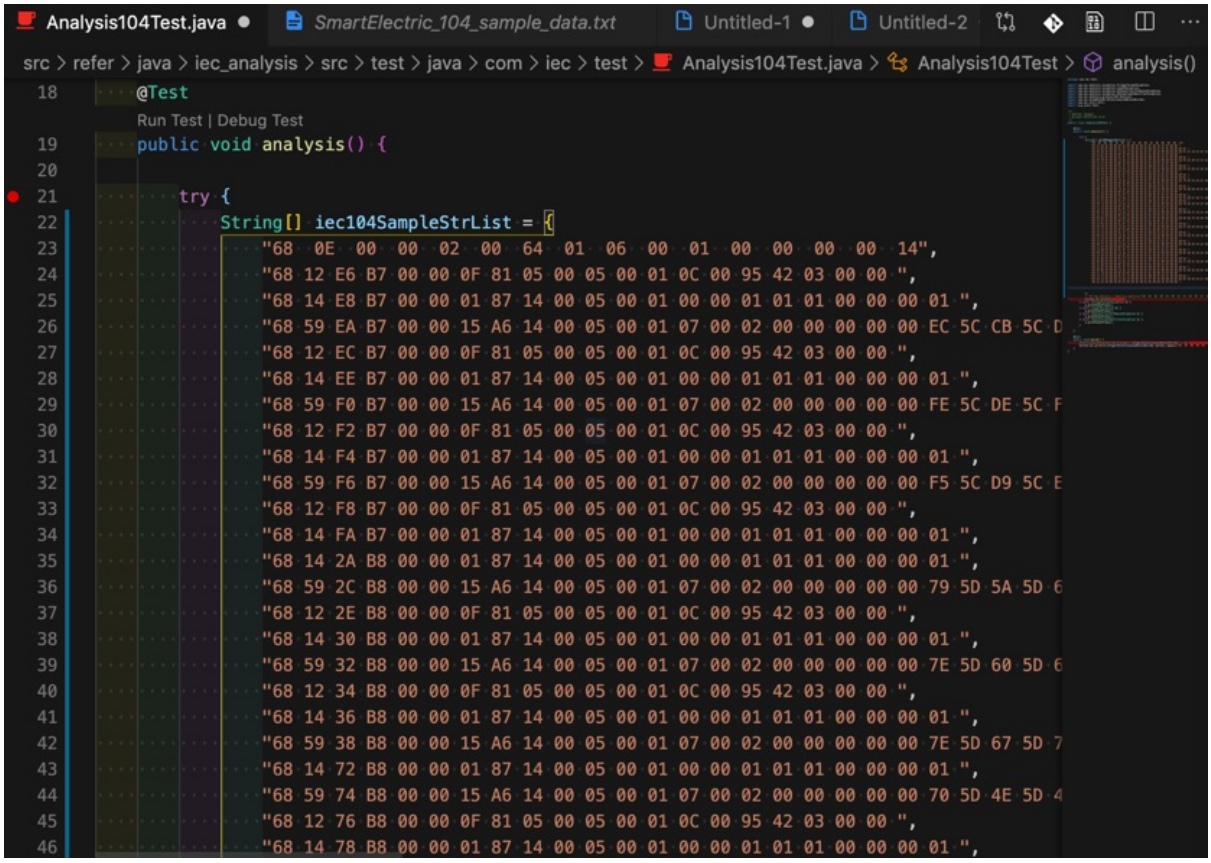


变成：




```
                "68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00 ",
                "68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01 ",
                "68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5
C DF 5C 45 00 01 00 FE FF 7C 00 94 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0 00 00 00 00 00 00 00 86 13 00 00 02 00 00 00 00 FE FF 00 00 00 00 87 00 00 00 00 0
0 02 00 00 00 00 00 ",
                "68 12 EC B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00 ",
...
                "68 12 16 D3 00 00 0F 81 05 00 05 00 01 0C 00 B6 42 03 00 00 ",
```

用于粘贴到代码中使用：

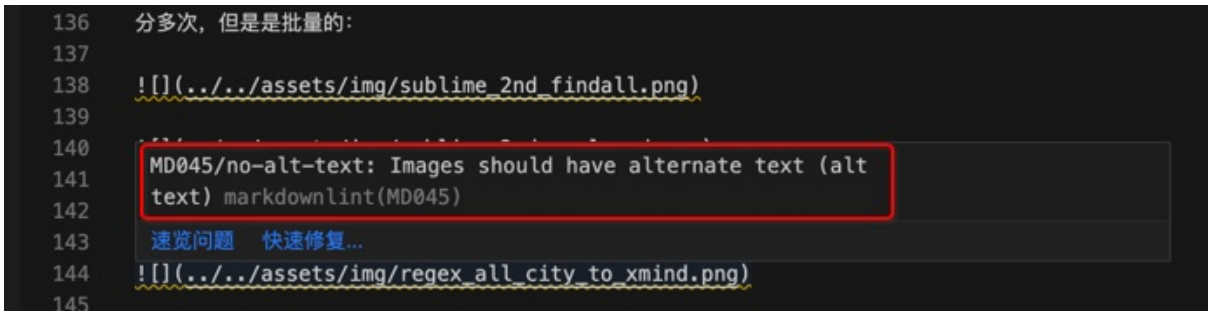


->从而把：

- 繁琐的，手工的，从原始文件中拷贝和粘贴的重复劳动，
- 快捷的，自动的，完成，且更准确，不会出现手动操作的失误。

填充Markdown中图片文件名

此处正在写教程期间，正好有个需求：为了避免和消除Markdown中的，关于图片的文件名即image的alt的text是空的警告：



然后正好用正则，去自动填充此处image的alt的text，即图片的文件名

用正则：

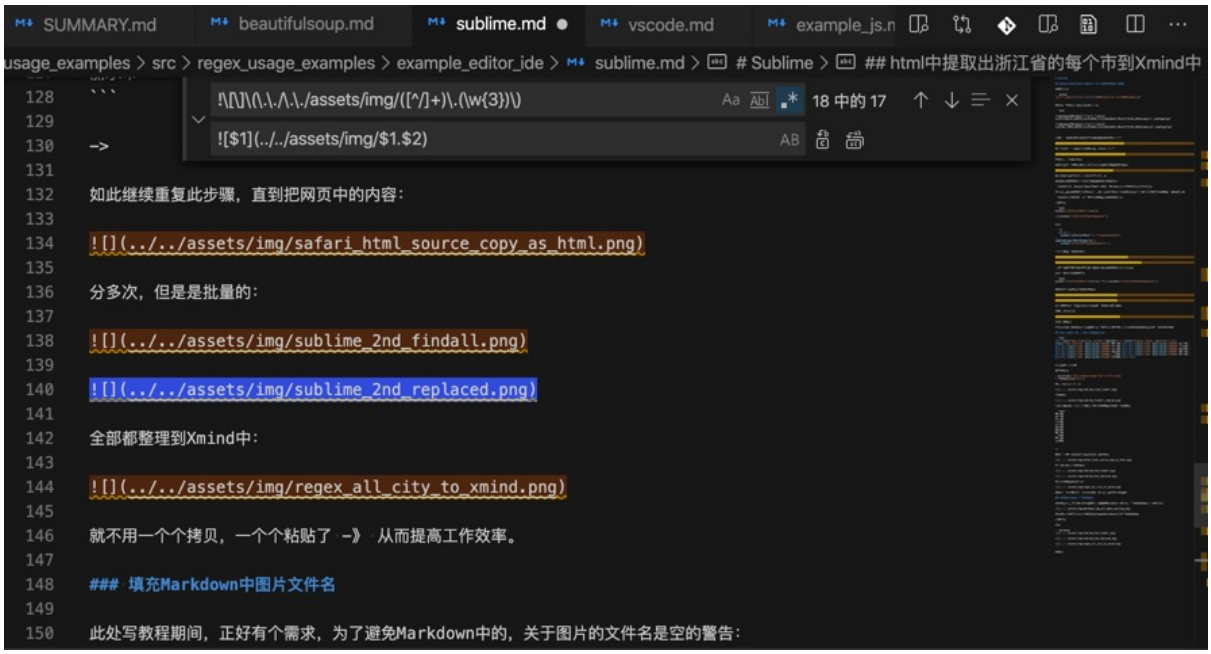
```
\[\]\(\.\./\.\./assets/img/([^/]+)\.(\w{3})\)
[$1](../../assets/img/$1.$2)
```

把：

```
[sublime_2nd_findall](../../assets/img/sublime_2nd_findall.png)

[sublime_2nd_replaced](../../assets/img/sublime_2nd_replaced.png)

[regex_all_city_to_xmind](../../assets/img/regex_all_city_to_xmind.png)
```

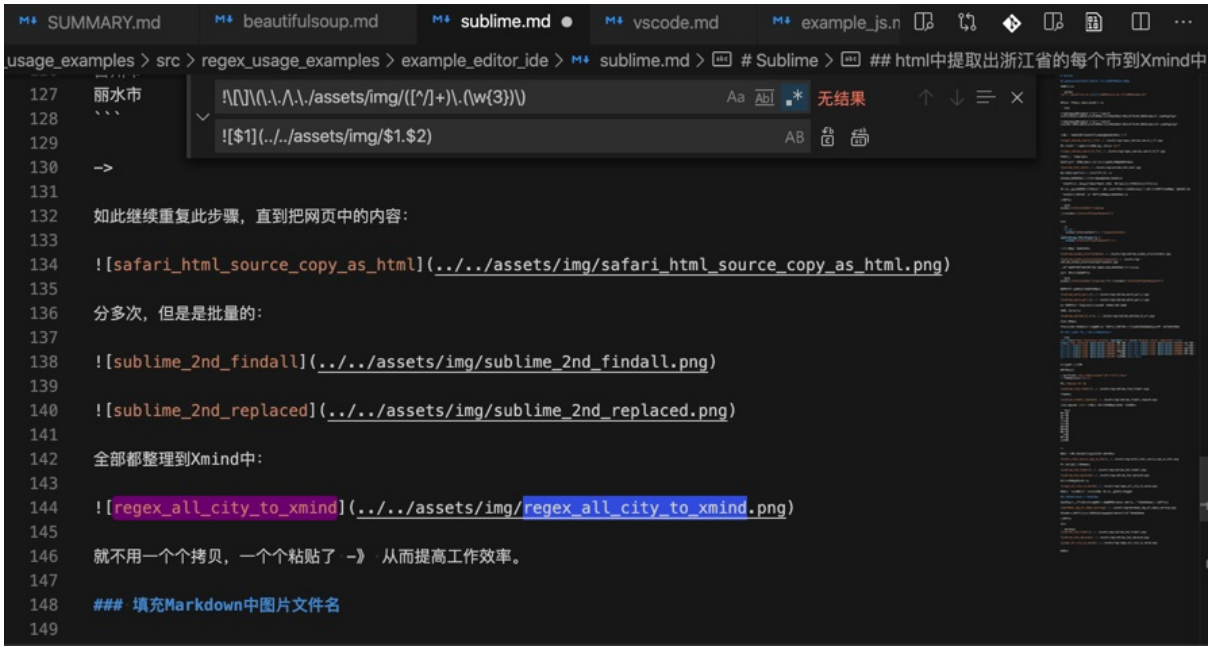


变成：

```
[sublime_2nd_findall](../../assets/img/sublime_2nd_findall.png)

[sublime_2nd_replaced](../../assets/img/sublime_2nd_replaced.png)

[regex_all_city_to_xmind](../../assets/img/regex_all_city_to_xmind.png)
```



即可自动填充image的alt的text，消除Markdown中的警告了。

附录

下面列出相关参考资料。

参考资料

- 【教程】BeautifulSoup中使用正则表达式去搜索多种可能的关键字 – 在路上
- 【已解决】VSCode中如何使用正则表达式去替换且被替换中使用分组group
- 【整理】中国常见的城市的名字