目录

正则表达式应用举例

- 最新版本： `v1.1`
- 更新时间： `20201223`

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，包括各种编程语言中的实例，比如Javascript、Python中的BeautifulSoup、数据库中的MongoDB和MongoDB Compass，以及编辑器中的实例，包括Sublime、以及对正则支持的很好也很好用的VSCode。

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

鸣谢

感谢我的老婆陈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。

更多其他电子书

本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

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：

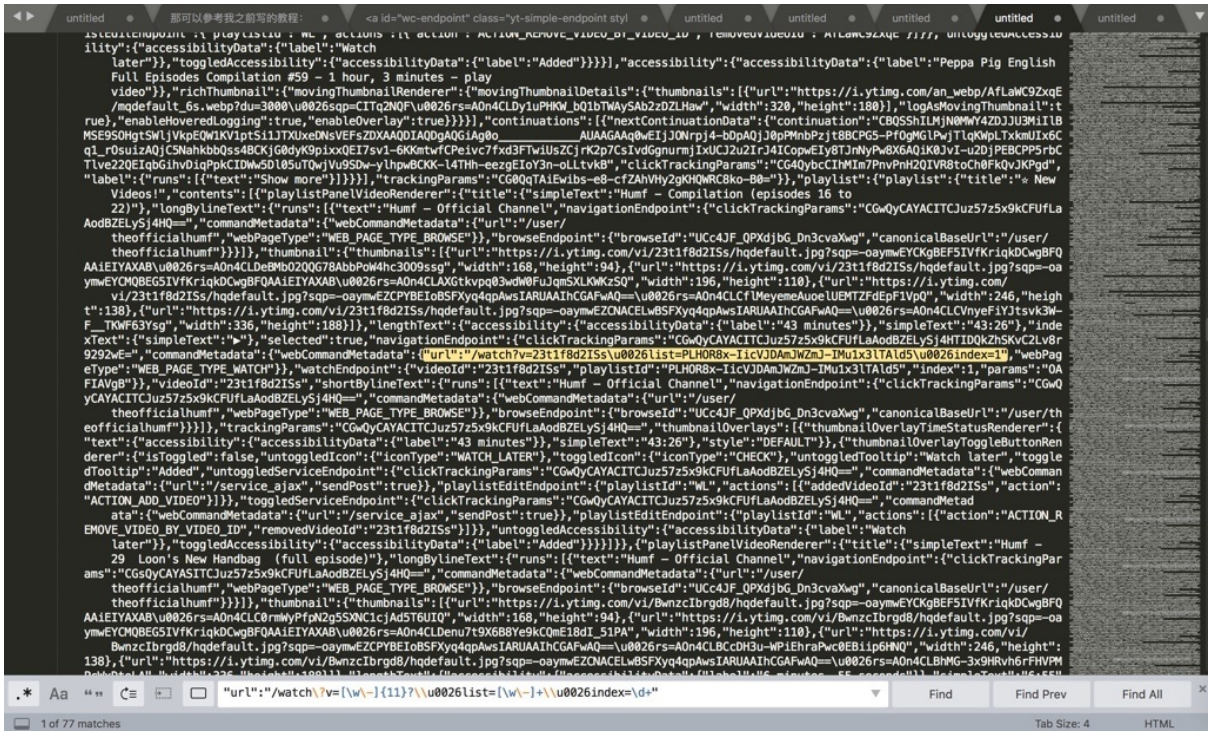
```
"url":"/watch\?v=[\w\-]{11}?\\u0026list=[\w\-]+\\u0026index=\d+"
```

可以从一堆的js中的script的值中：

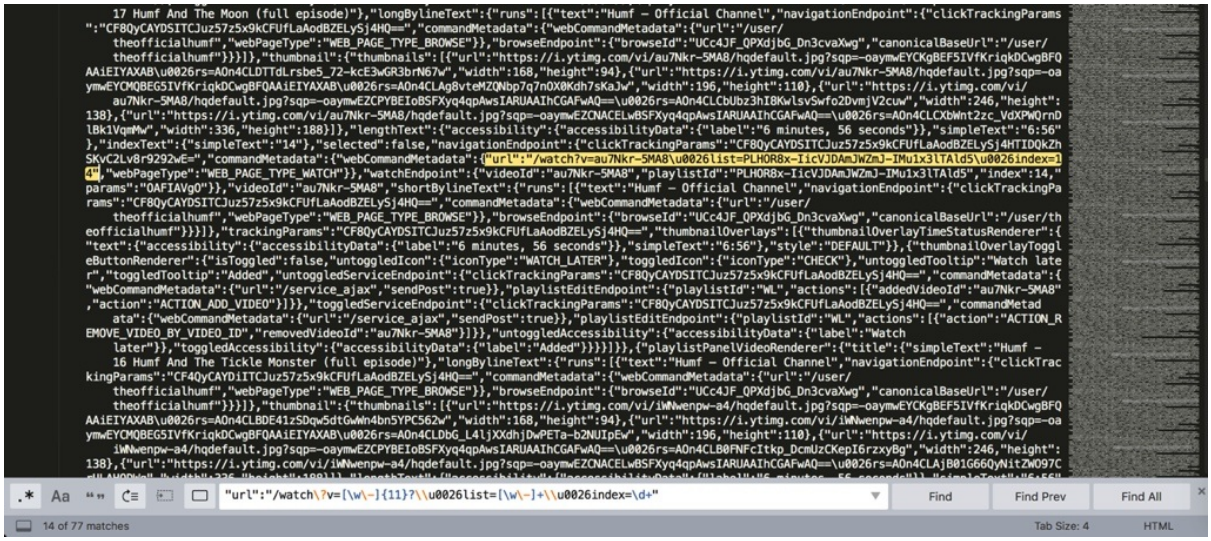
```
{"webCommandMetadata":{"url":"/watch?v=23t1f8d2ISs\u0026list=PLHOR8x-IicVJDAmJWZmJ-IMu1x3lTAld5\u0026index=1",”webPageType"

{"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`

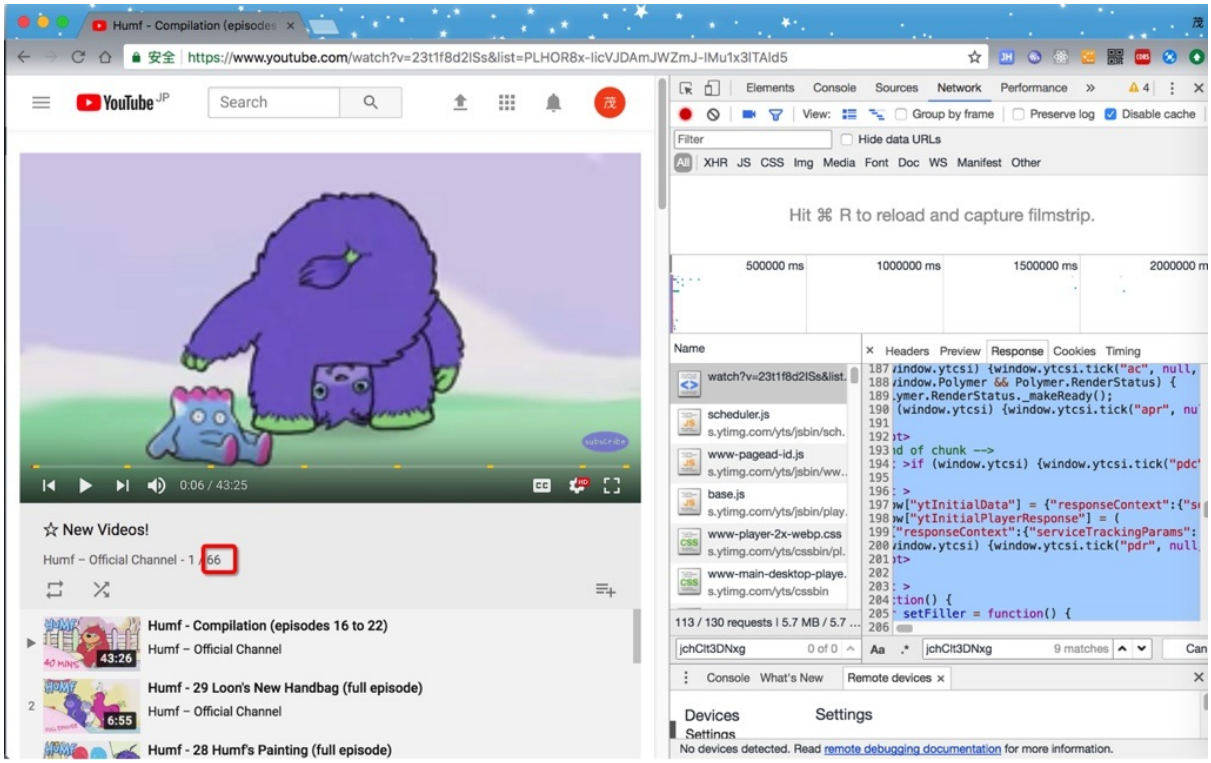


点击Find，继续向下找，比如找到第14个： `14/77`



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：

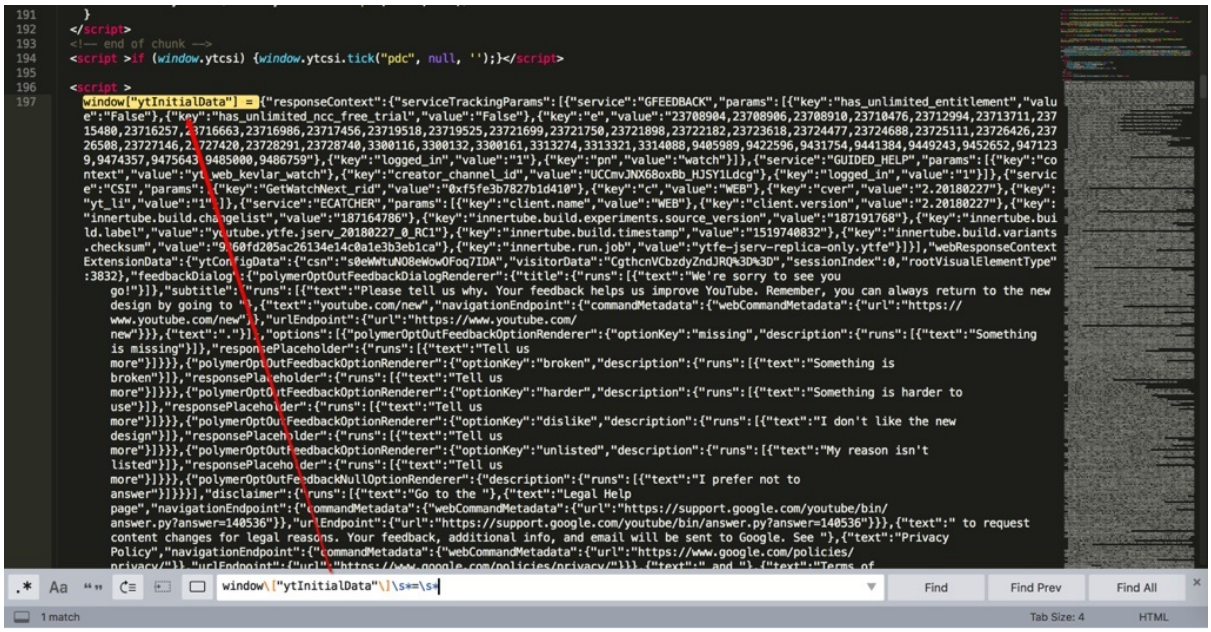
```
window\["ytInitialData"\]\s* \s*

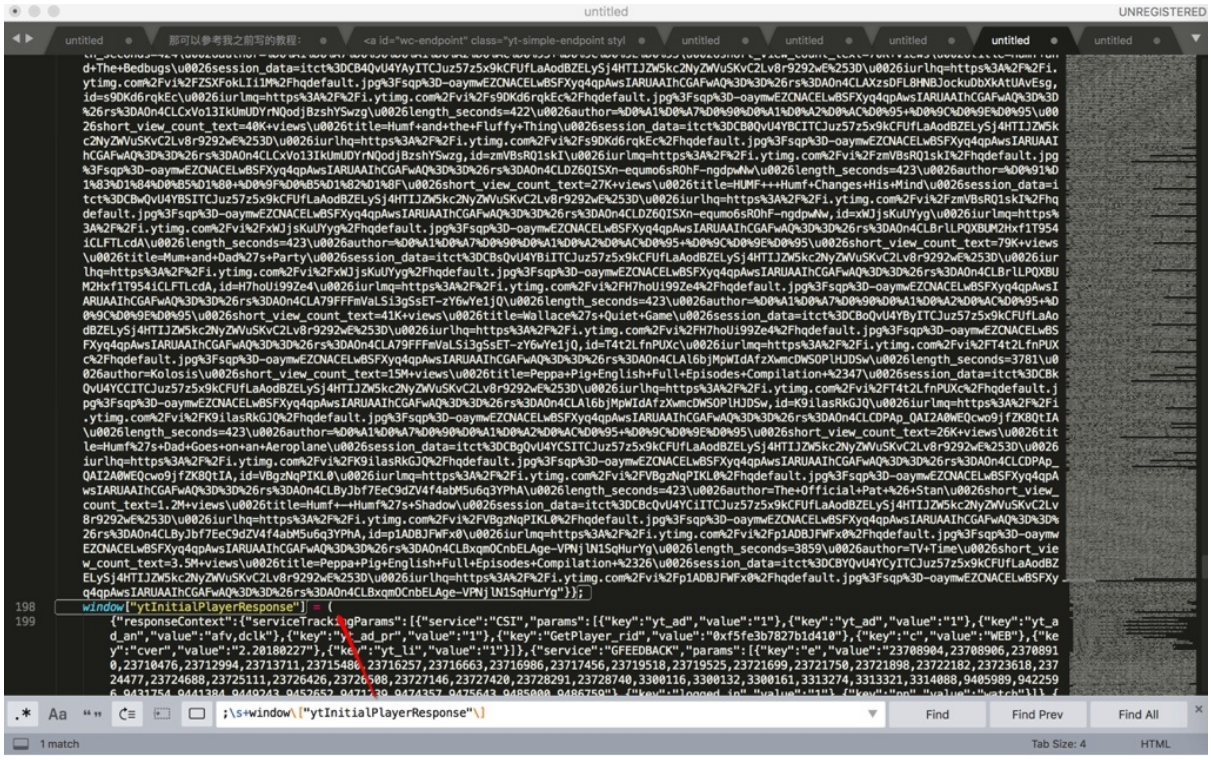
;\s+window\["ytInitialPlayerResponse"\]
```

从：

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

中，搜索到所要的内容：



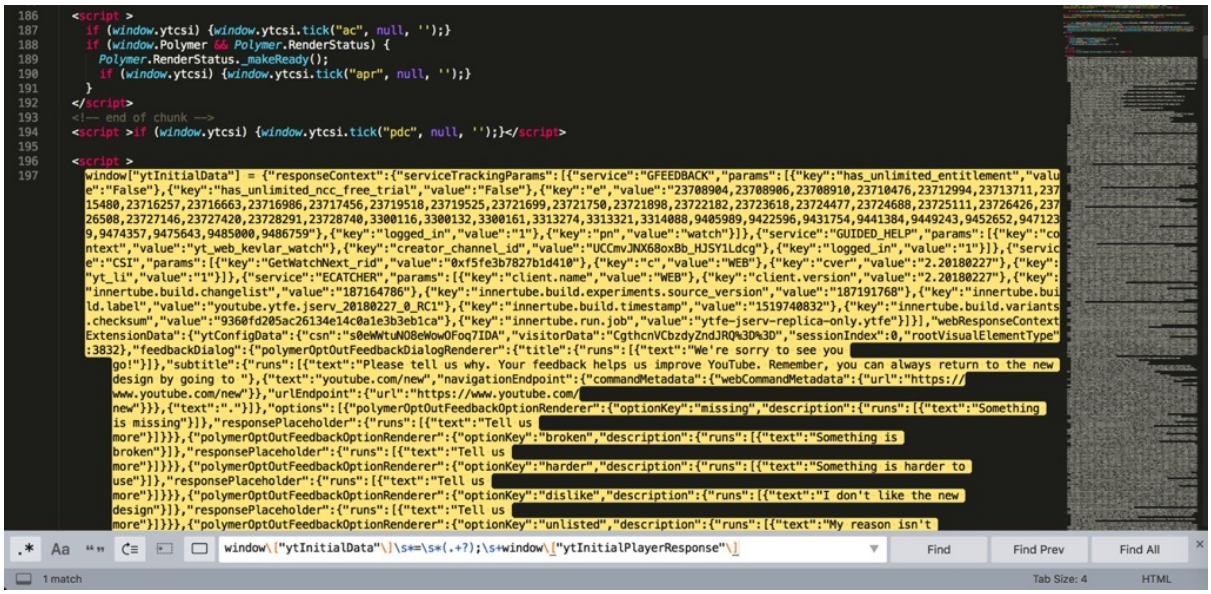


而首尾的正则之间的内容，就是需要找的js的变量的值，是个json

对应着写个完整的正则：

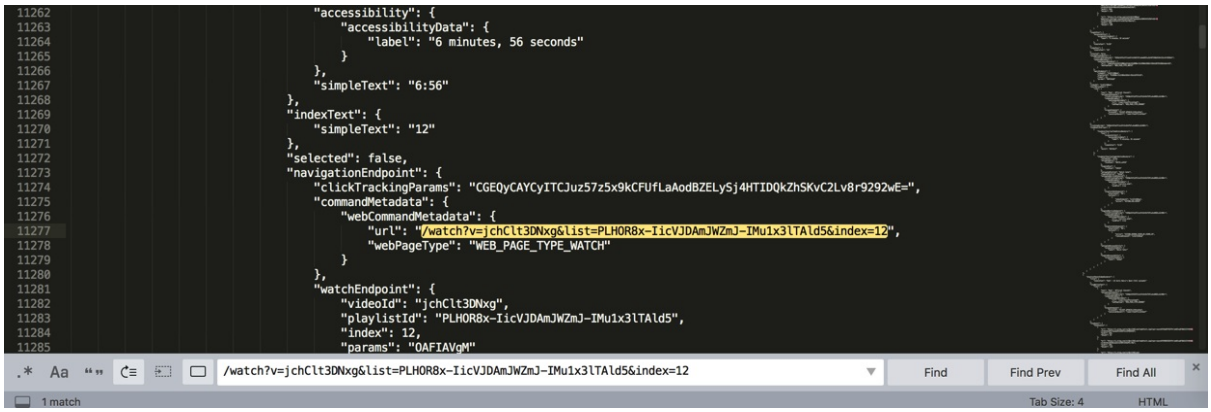
```
window\["ytInitialData"\]\s*=\s*(.+?);\s+window\["ytInitialPlayerResponse"\]
```

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

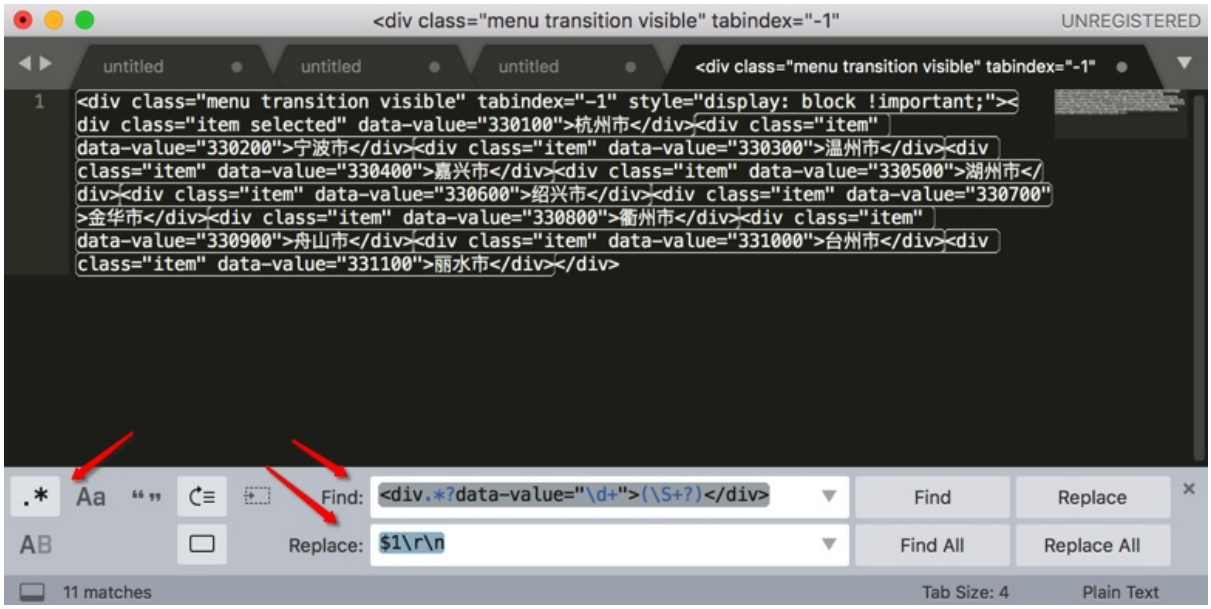
```
<div class="menu transition visible" tabindex="-1" style="display: block !important;"><div
class="item selected" data-value="330100">杭州市</div><div class="item" data-value="330200">
宁波市</div><div class="item" data-value="330300">温州市</div><div class="item" data-value="
330400">嘉兴市</div><div class="item" 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

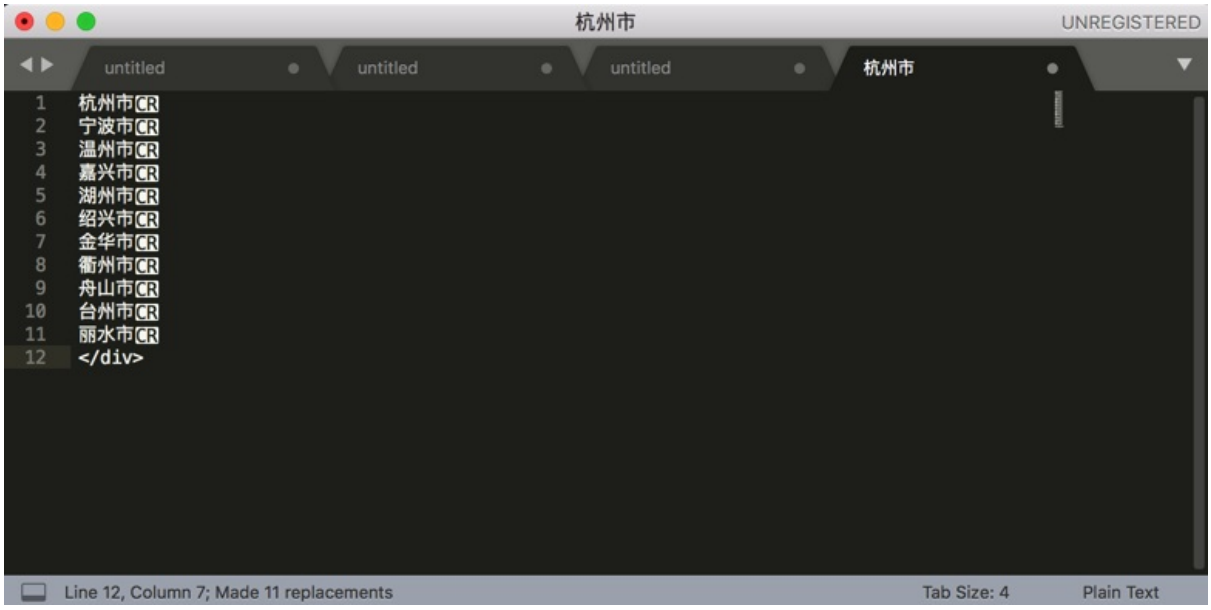
正则写法：

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

点击 `Replace All` ：



替换成：



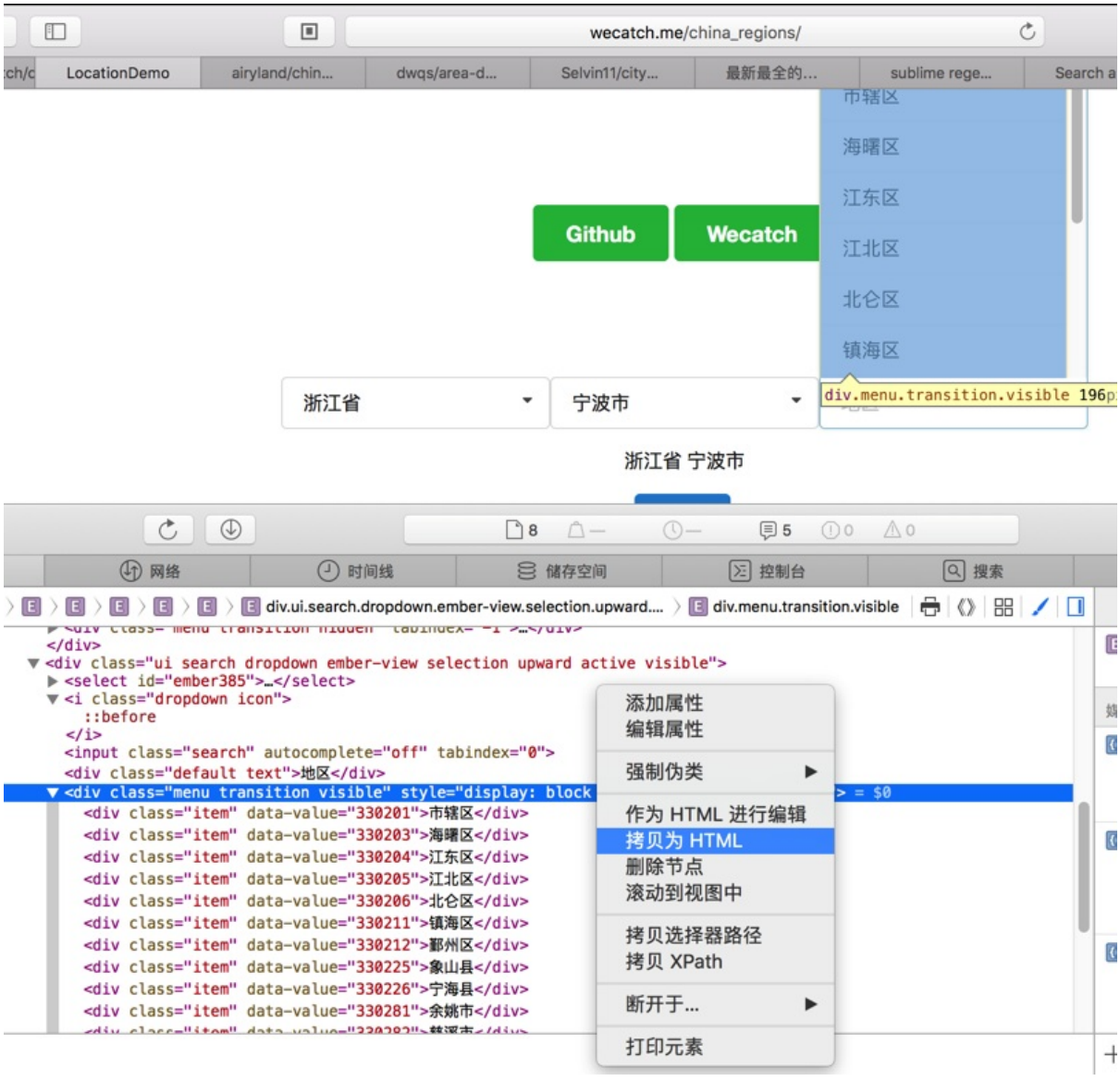
忽略掉最后的 `</div>` ，拷贝出来，即可得到我要的所有的市：

```
杭州市
宁波市
温州市
嘉兴市
湖州市
```

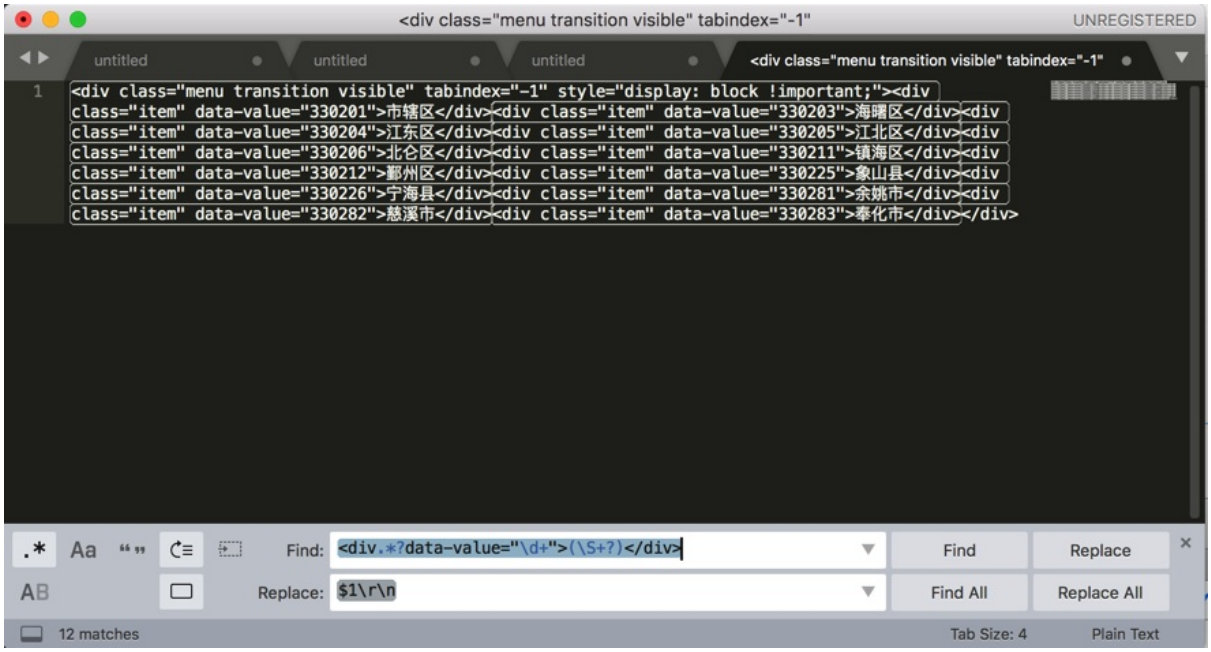
```
绍兴市
金华市
衢州市
舟山市
台州市
丽水市
```

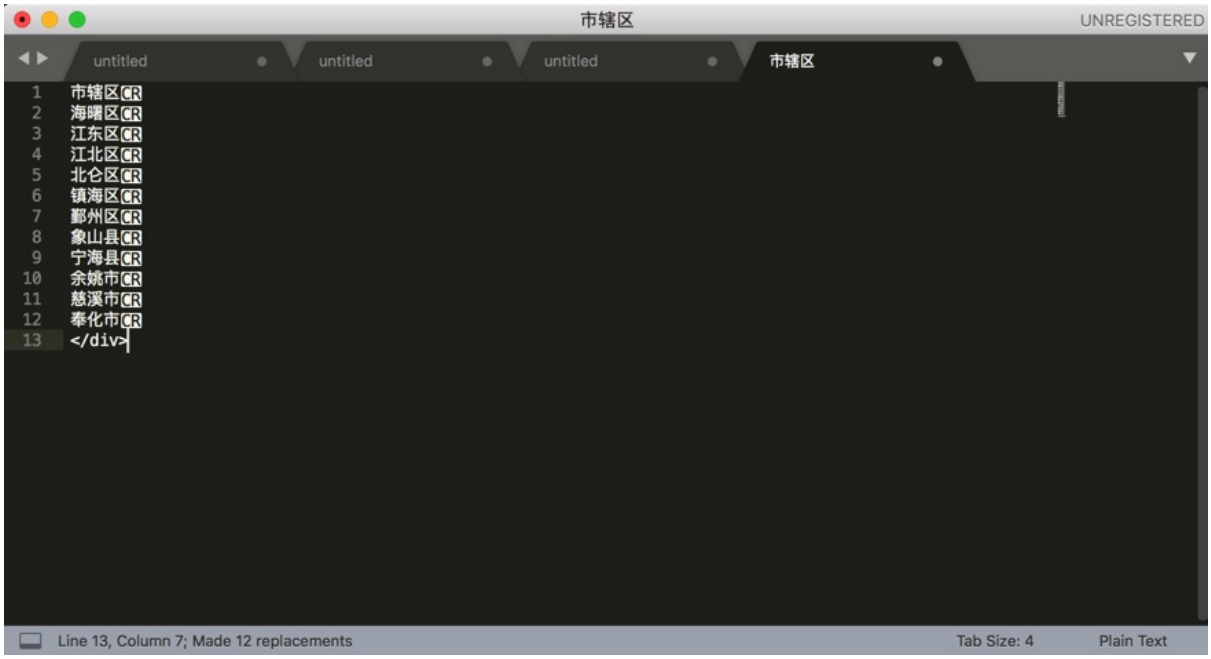
->

如此继续重复此步骤，直到把网页中的内容：

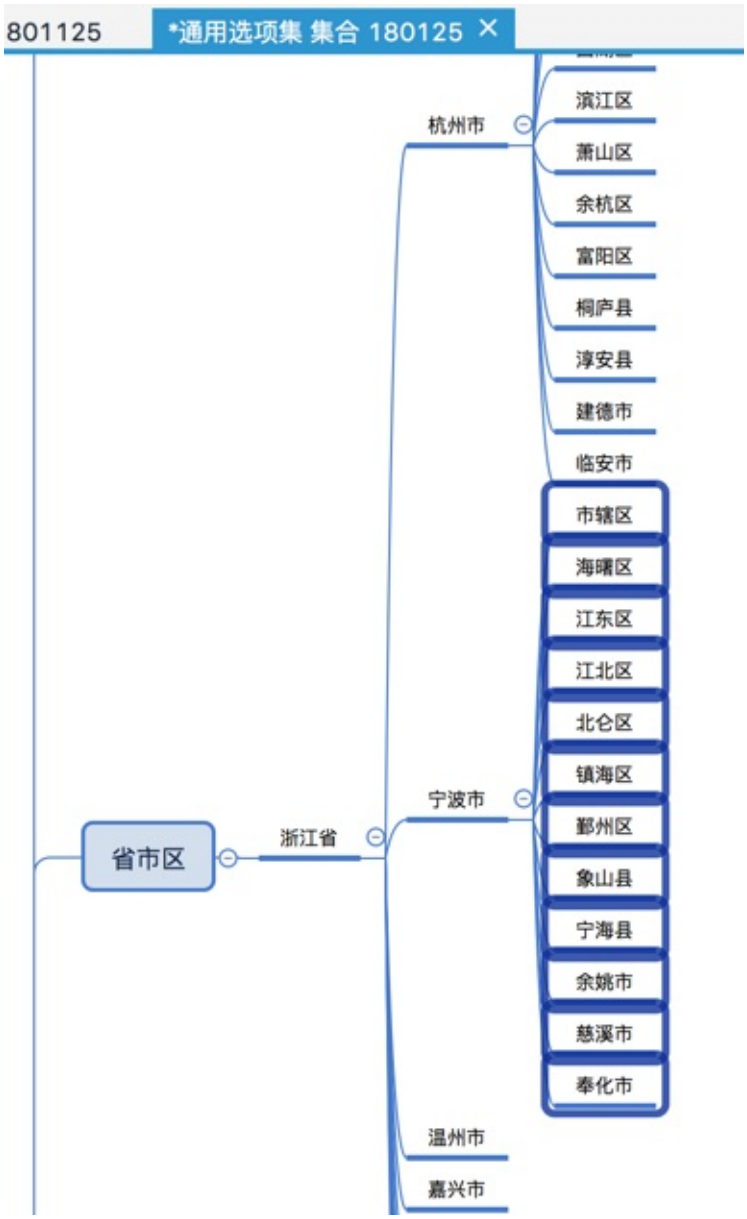


分多次，但是是批量的：





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

VSCode中正则语法说明

- 替换中引用搜索中的分组
  - 背景：搜索带 `分组`
    - 写法：`(xxx)`
  - 语法：替换时引用对应分组，用：`$N`
    - N=1,2,3,...
  - 举例
    - 例子
      - 搜索：`(\w+)\n`
      - 替换：`"$1",\n`
- 环视
  - 可以参考自己的教程
    - 环视断言 · 应用广泛的超强搜索：正则表达式
  - 核心要点和举例
    - `(?=xxx): (positive) look ahead (assertion)=正向肯定断言`
    - `(?!xxx): negative look ahead (assertion)=正向否定断言`
    - `(?<=xxx): (positive) look behind (assertion)=反向肯定断言`
      - 例子1：

```
(?<=\s)([\.\w]+)
lt;$1 gt;
```

    - `(?<!xxx): negative look behind (assertion)=反向否定断言`
      - 例子1

```
(?<!http\.)get\(
```

      - 例子2

```
(?<!")15(?!")
```

一些大的例子

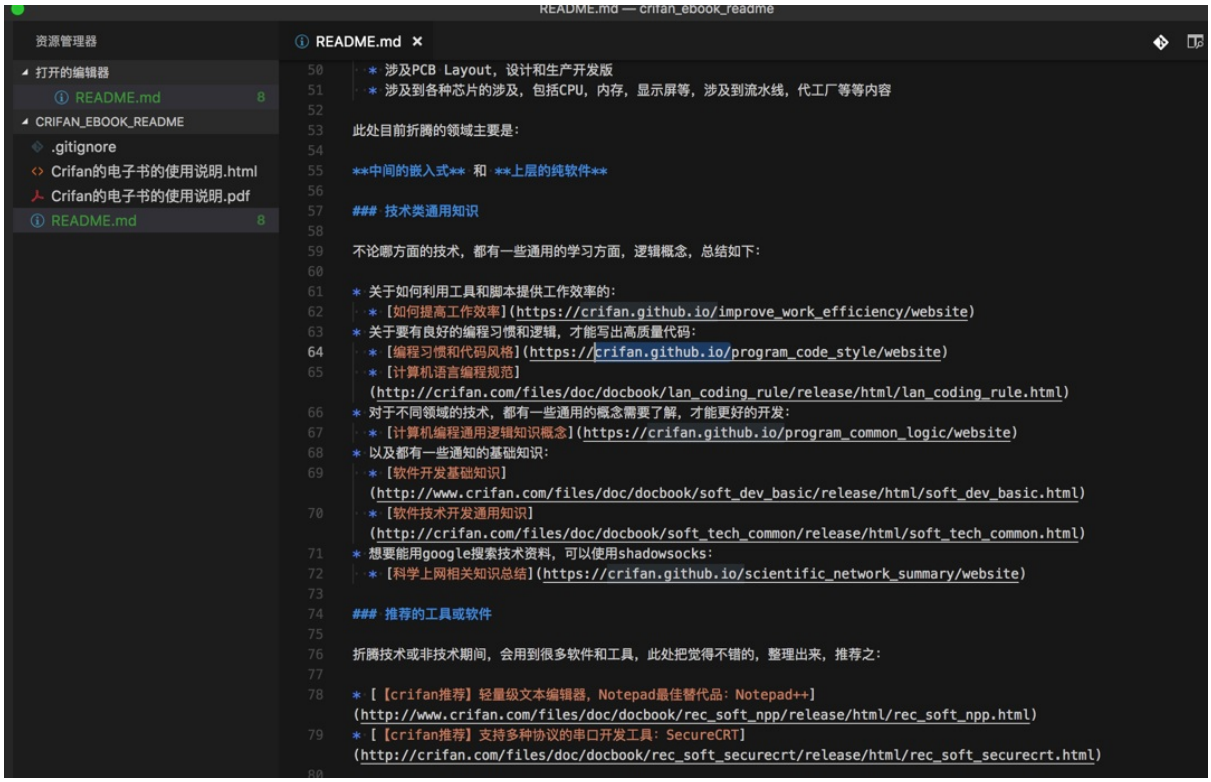
此处整理相对来说算是大的完整的一些的应用案例，供参考。

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`

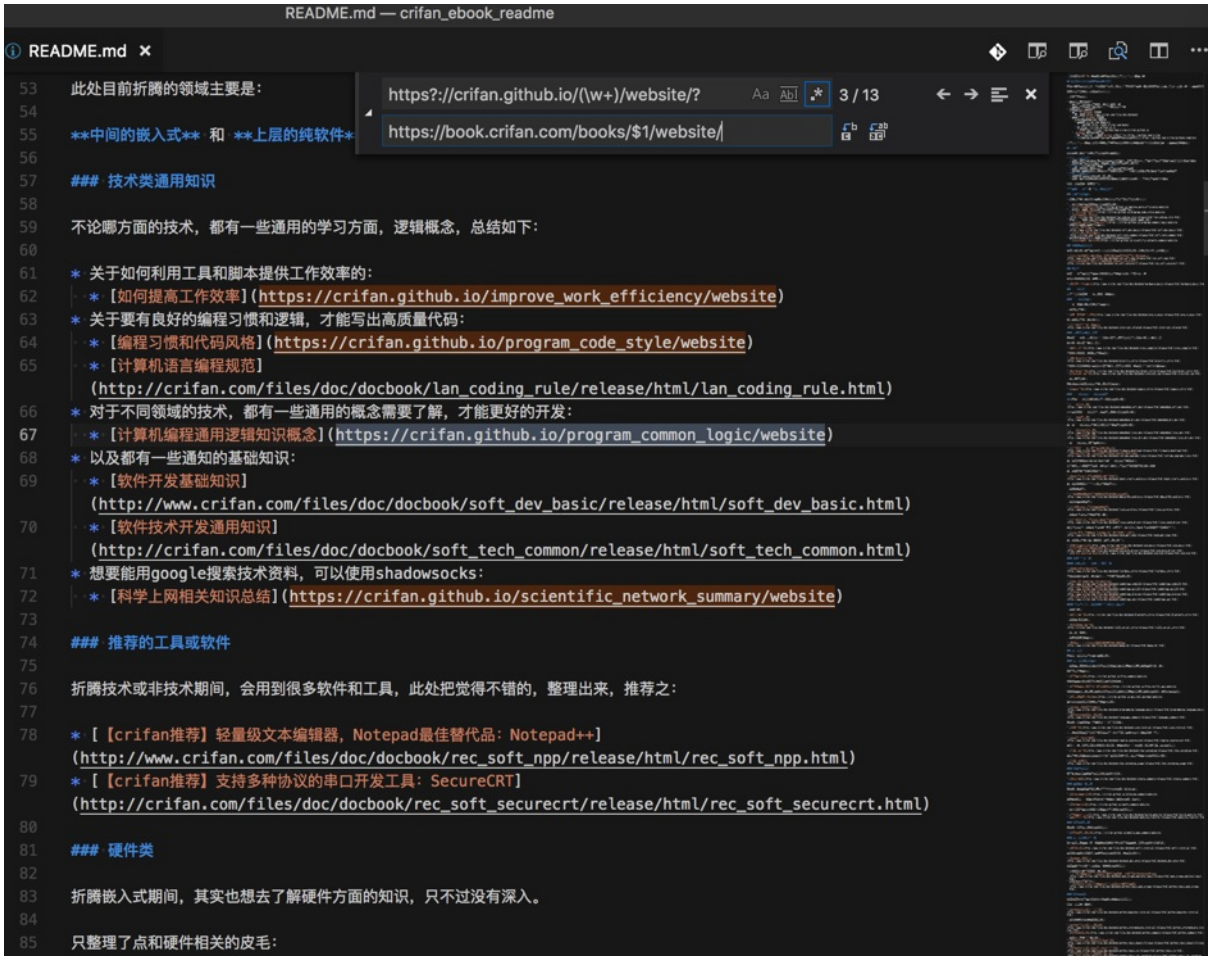
替换成：

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

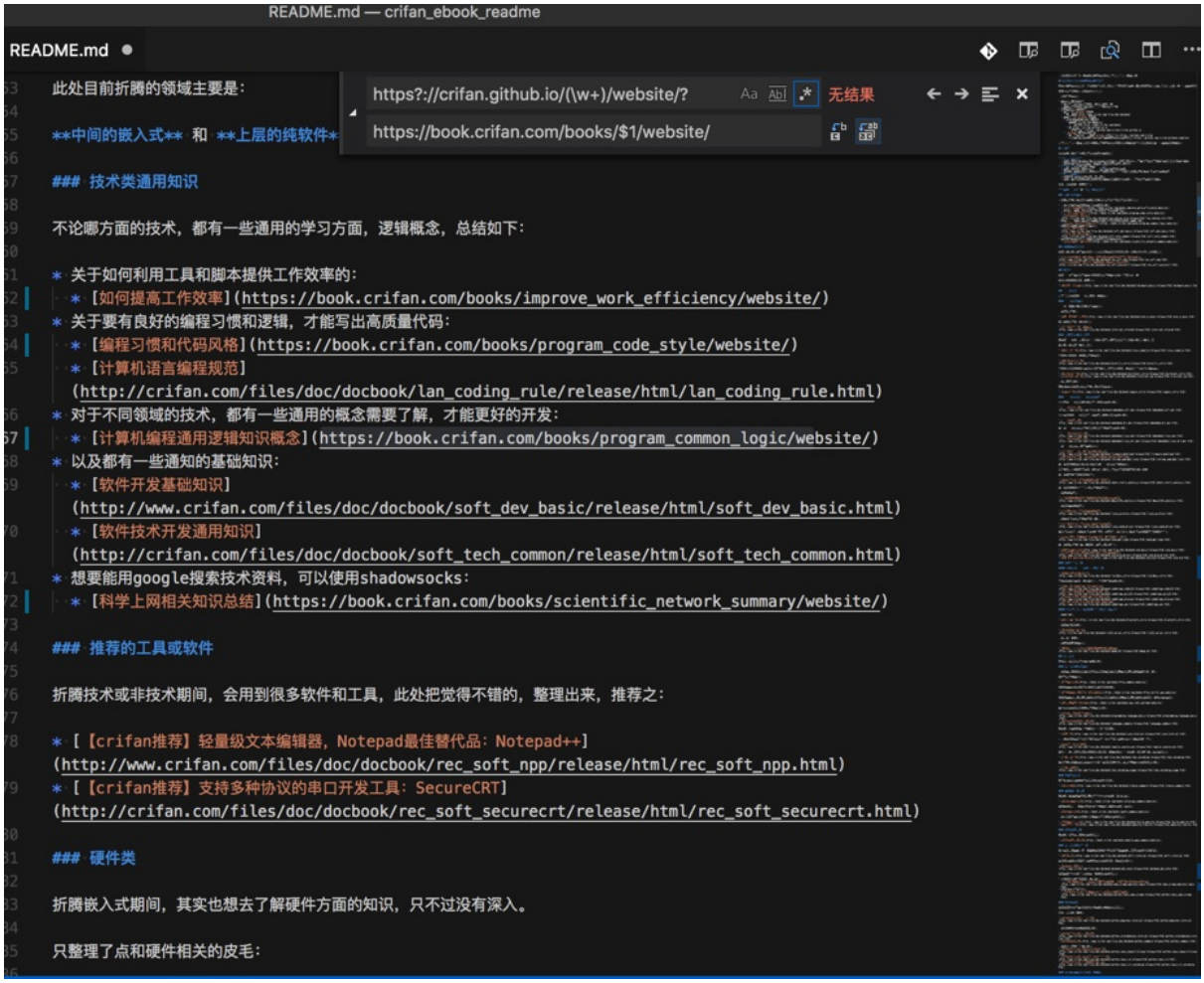
用正则：

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

实现从：



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
```

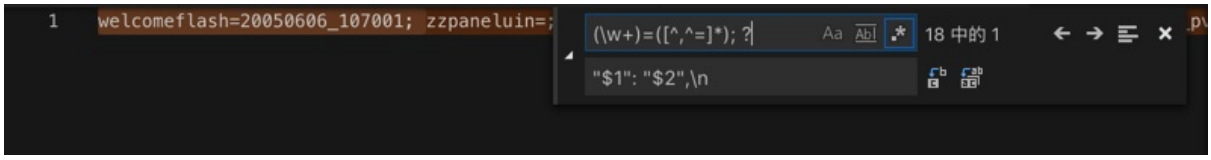
最后加上分号：

```
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;
```

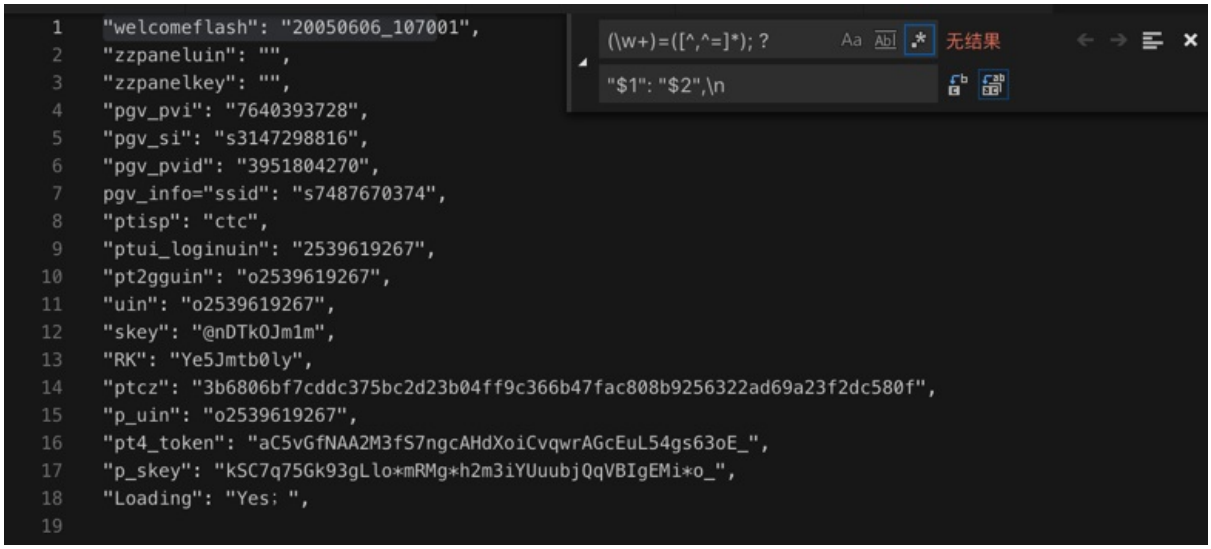
再去用正则：

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

替换：



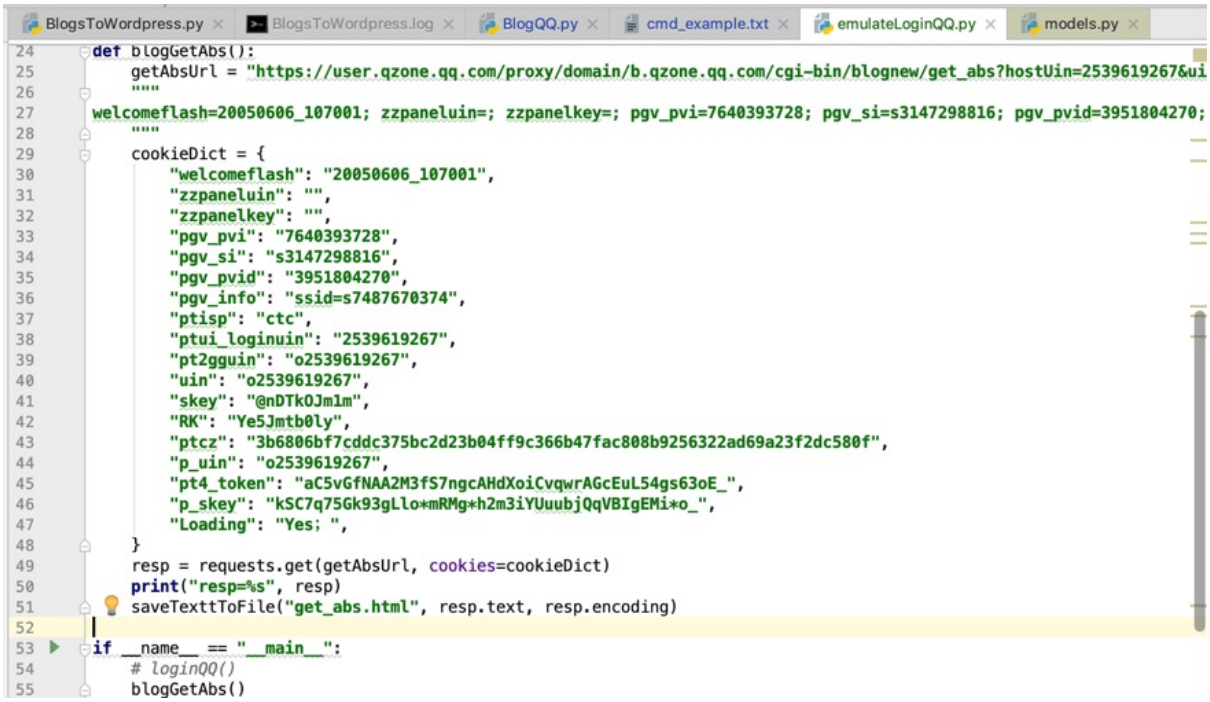
成自己要的dict的内容：



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：

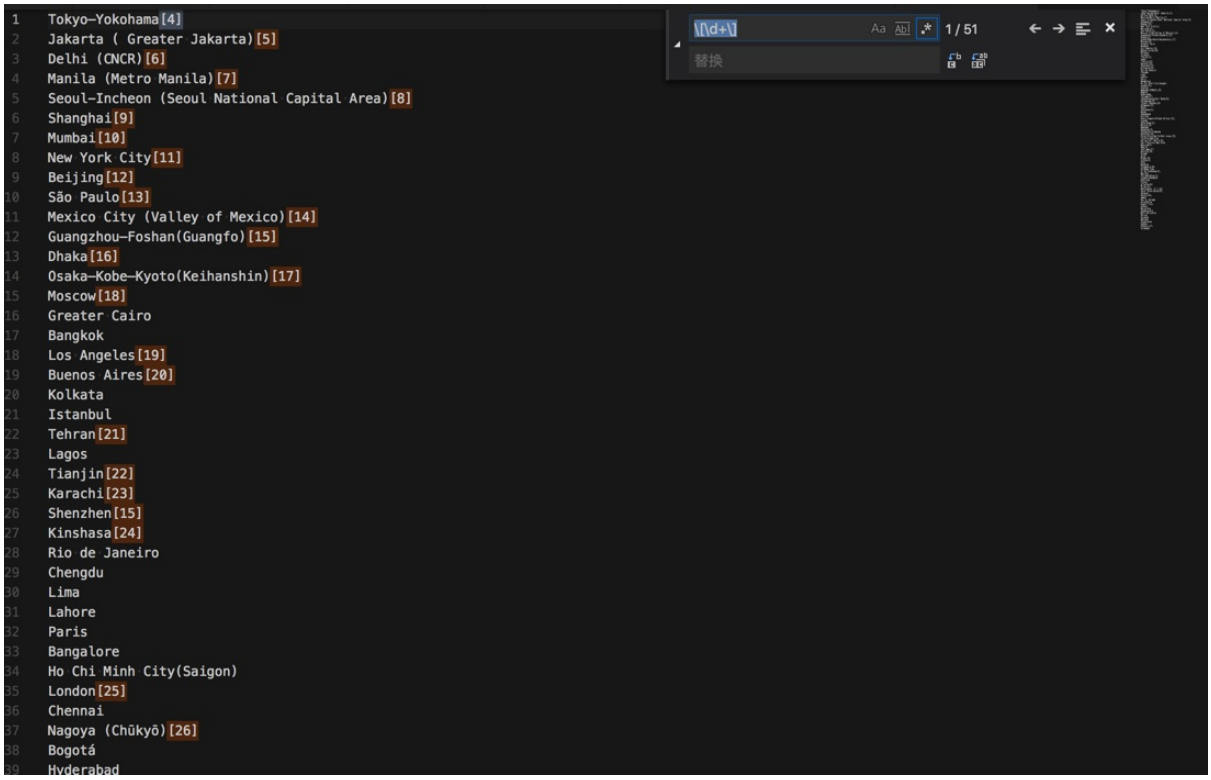
https://en.wikipedia.org/wiki/List_of_urban_areas_by_population

中的城市名，用正则：

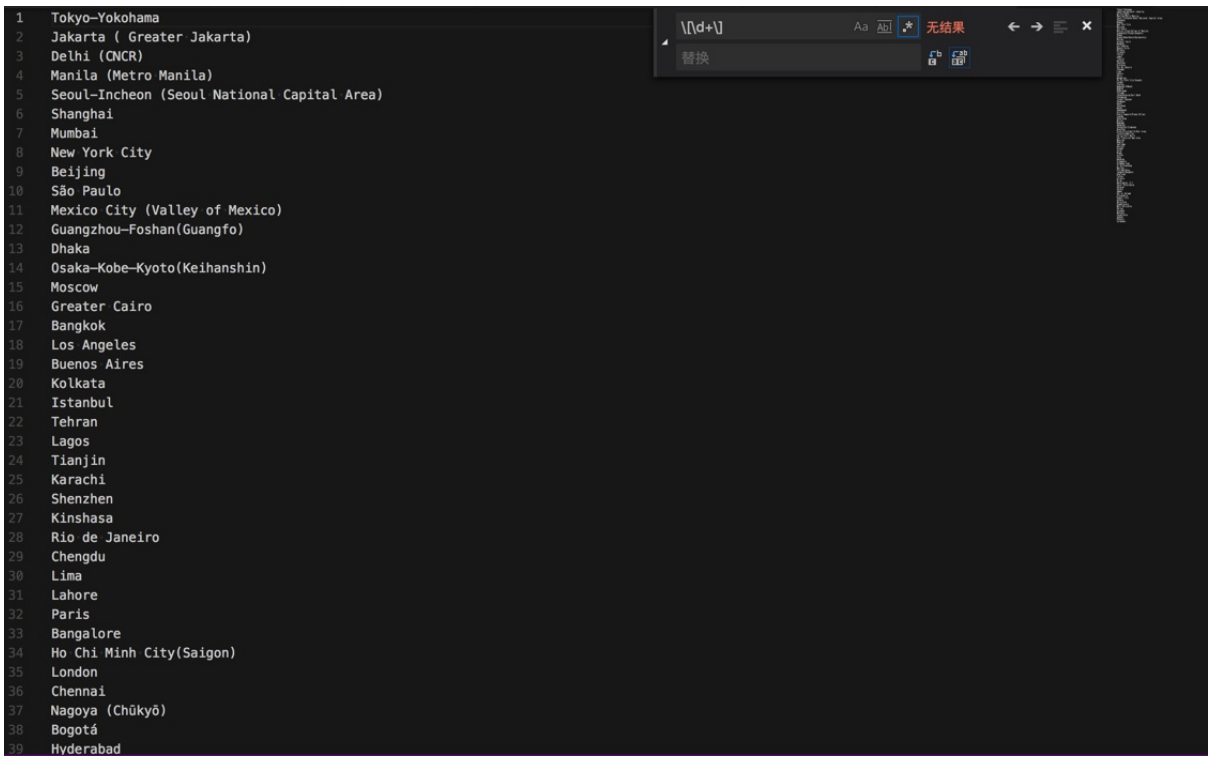
```
\[\d+\]
```

去除掉`[数字]`

从：



替换成：

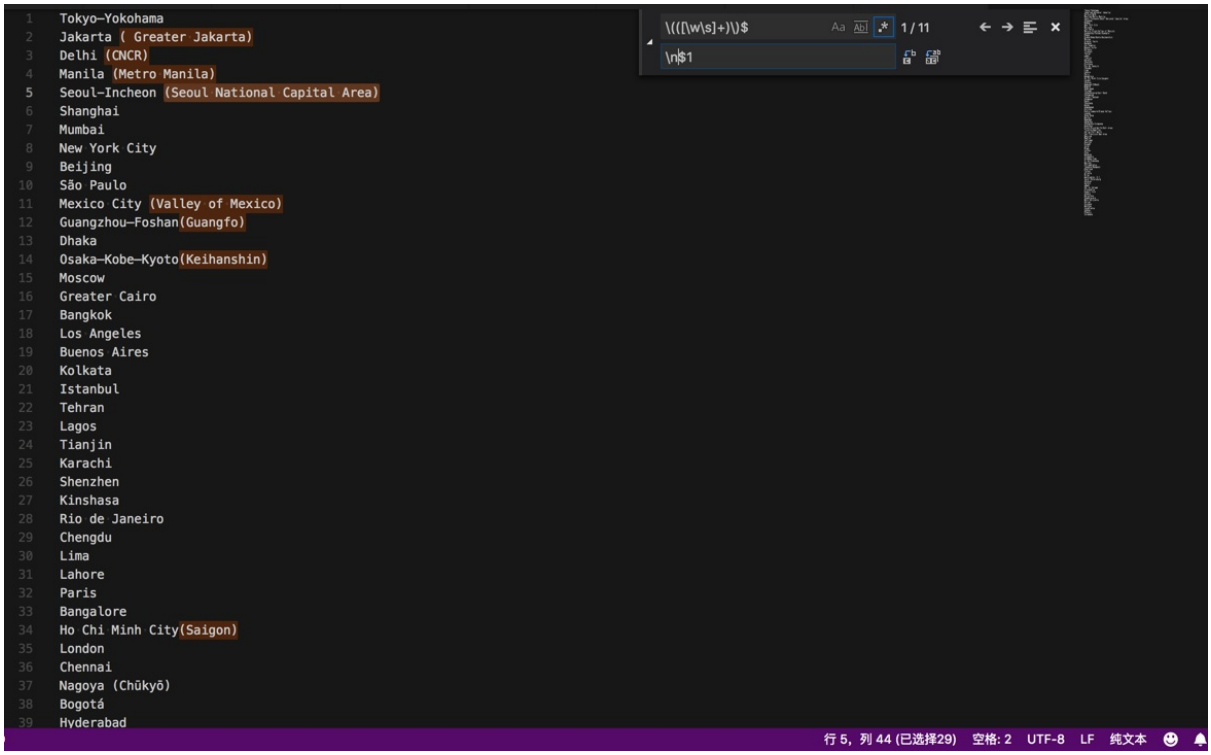


以及继续用：

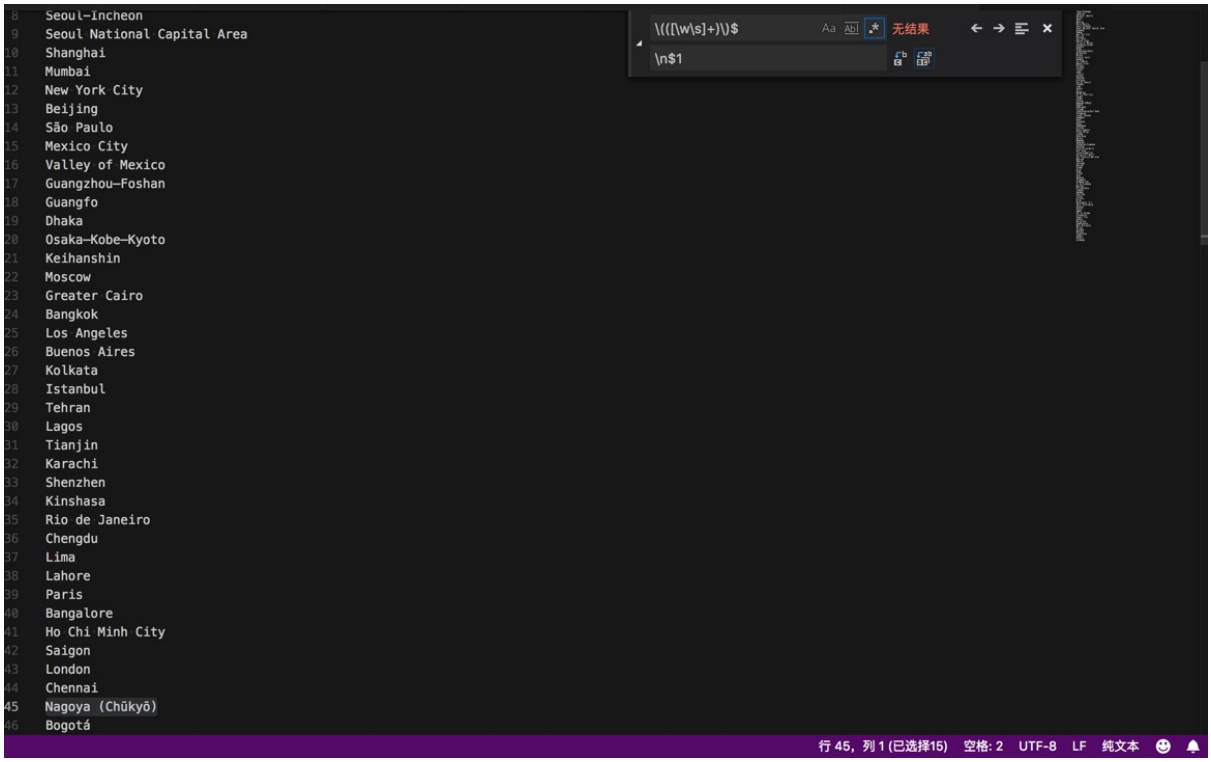
```
\(([\w\s]+)\)$
$1
```

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：

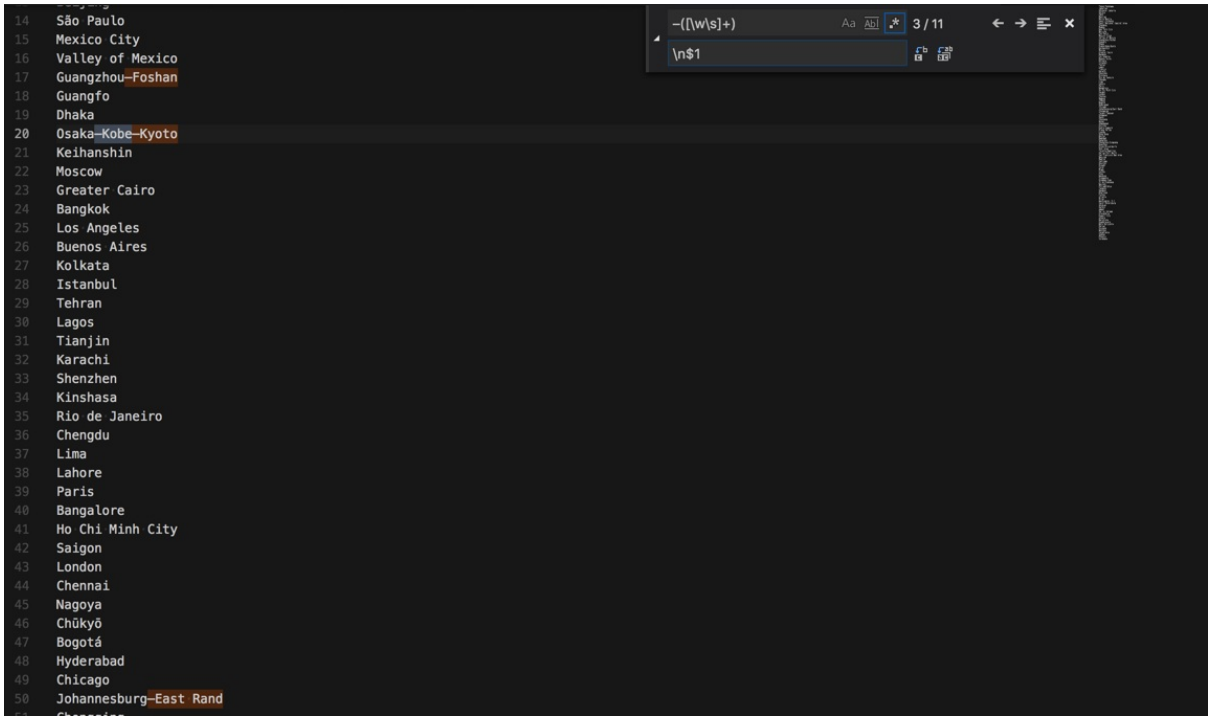
`Nagoya (Chūkyō)`

是因为里面有unicode的字符，由于数量不多，手动处理即可。

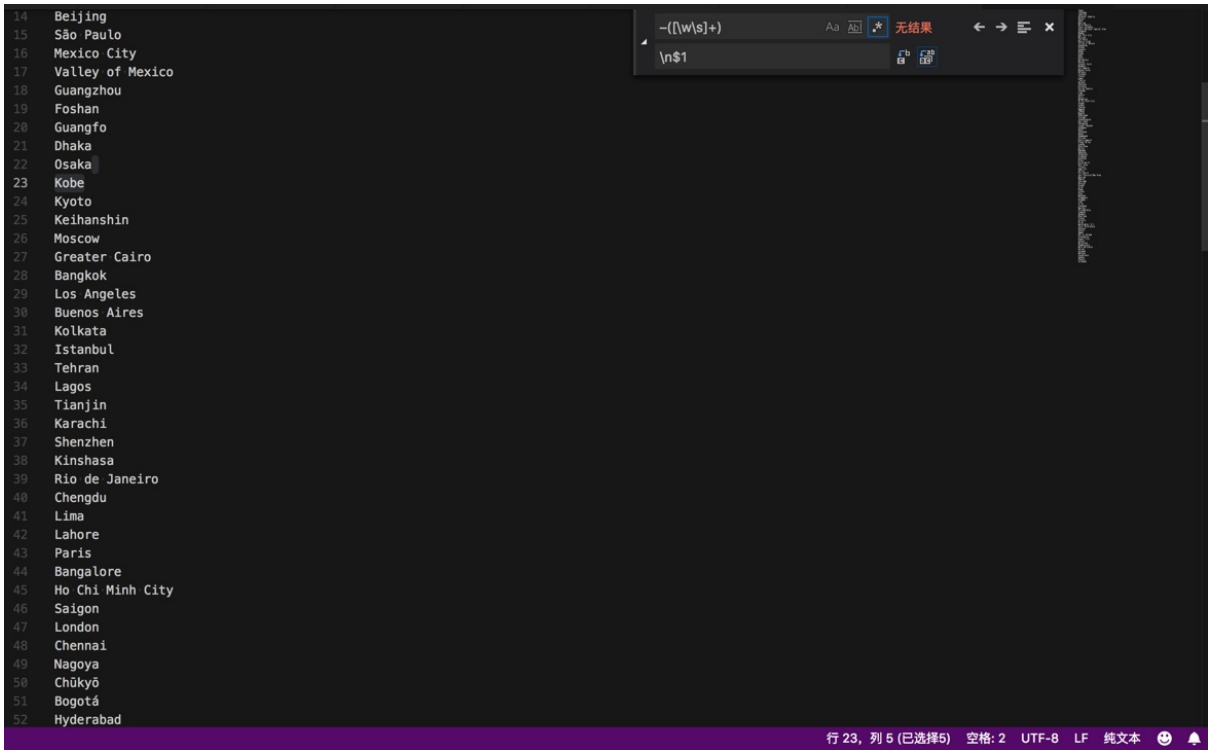
再去用：

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

把 `xxx-yyy` 中的 `yyy` 放到下一行



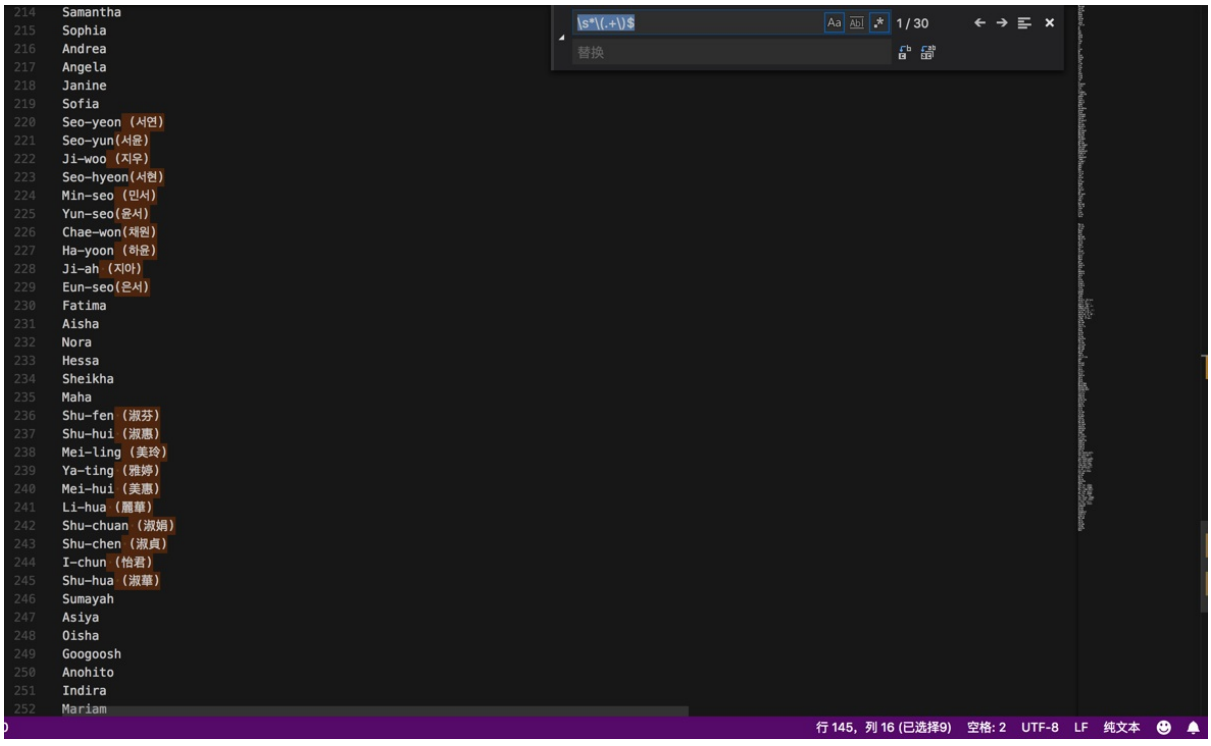
替换成：



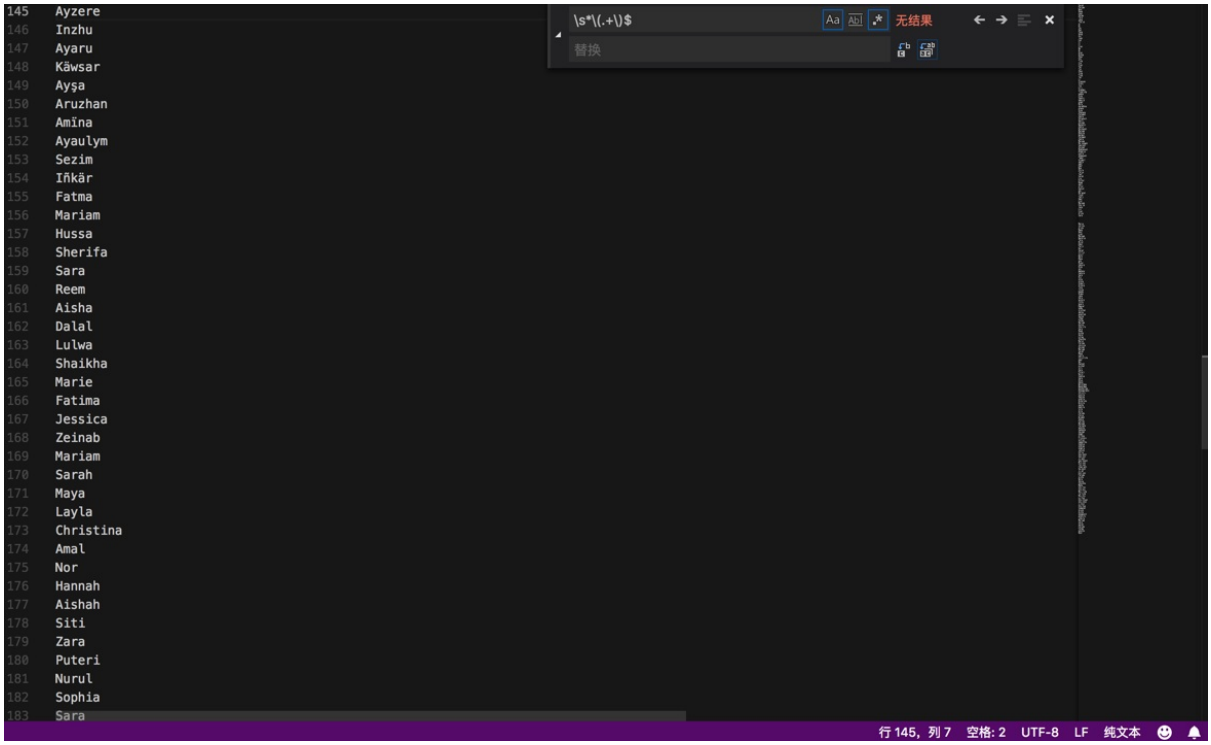
以及，用：

```
\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

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
```

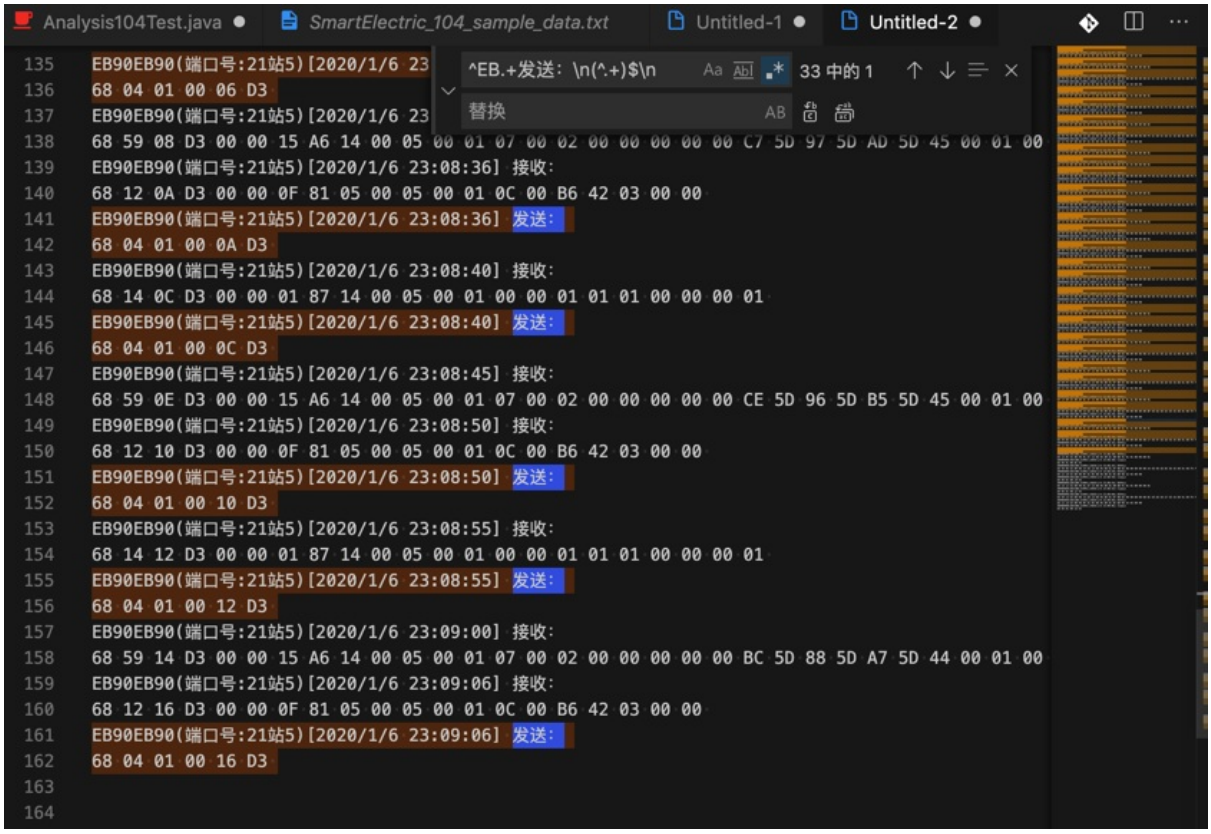
```
EB90EB90(端口号:21站5)[2020/1/6 18:19:12] 接收:
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
EB90EB90(端口号:21站5)[2020/1/6 18:19:12] 发送:
68 04 01 00 E8 B7
EB90EB90(端口号:21站5)[2020/1/6 18:19:17] 接收:
68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5C DF 5C 45 00 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

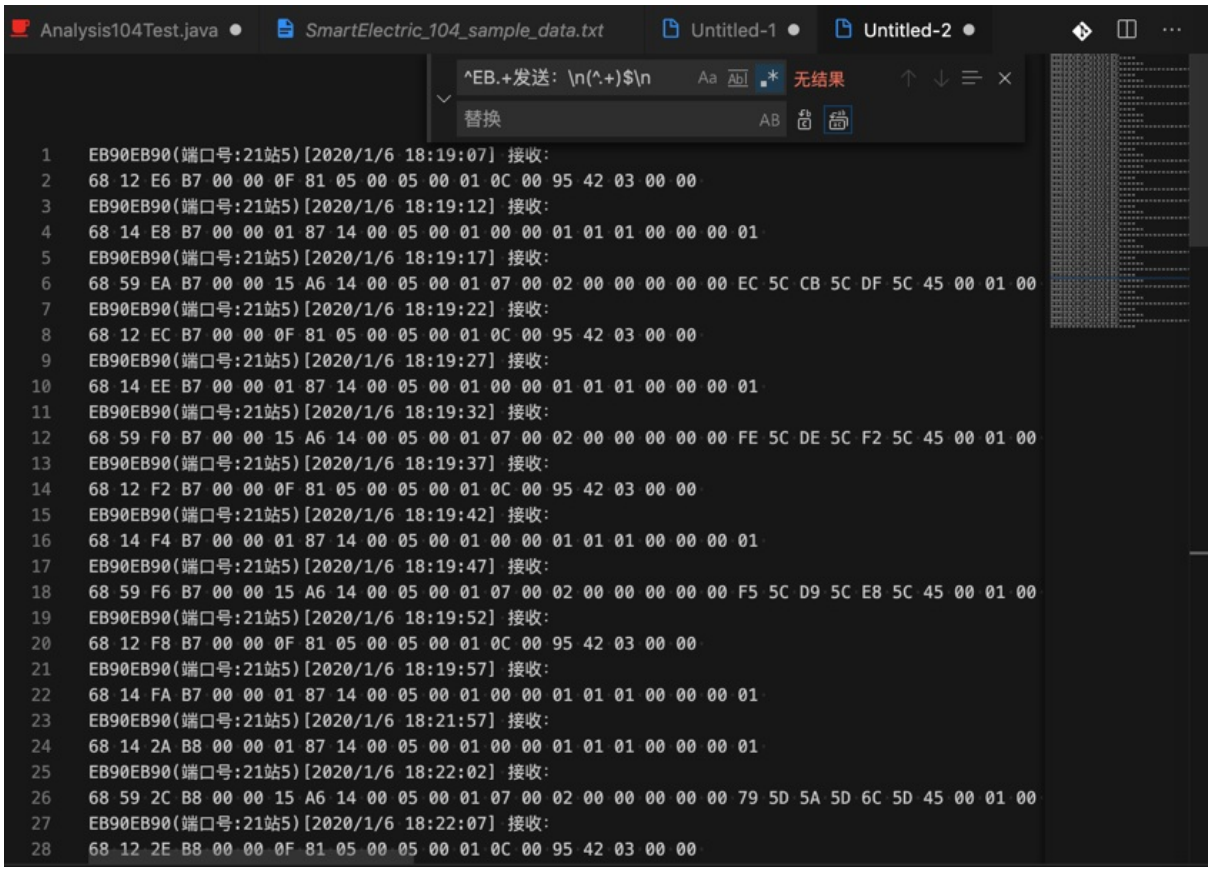
（1）先 去除 发送的部分

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

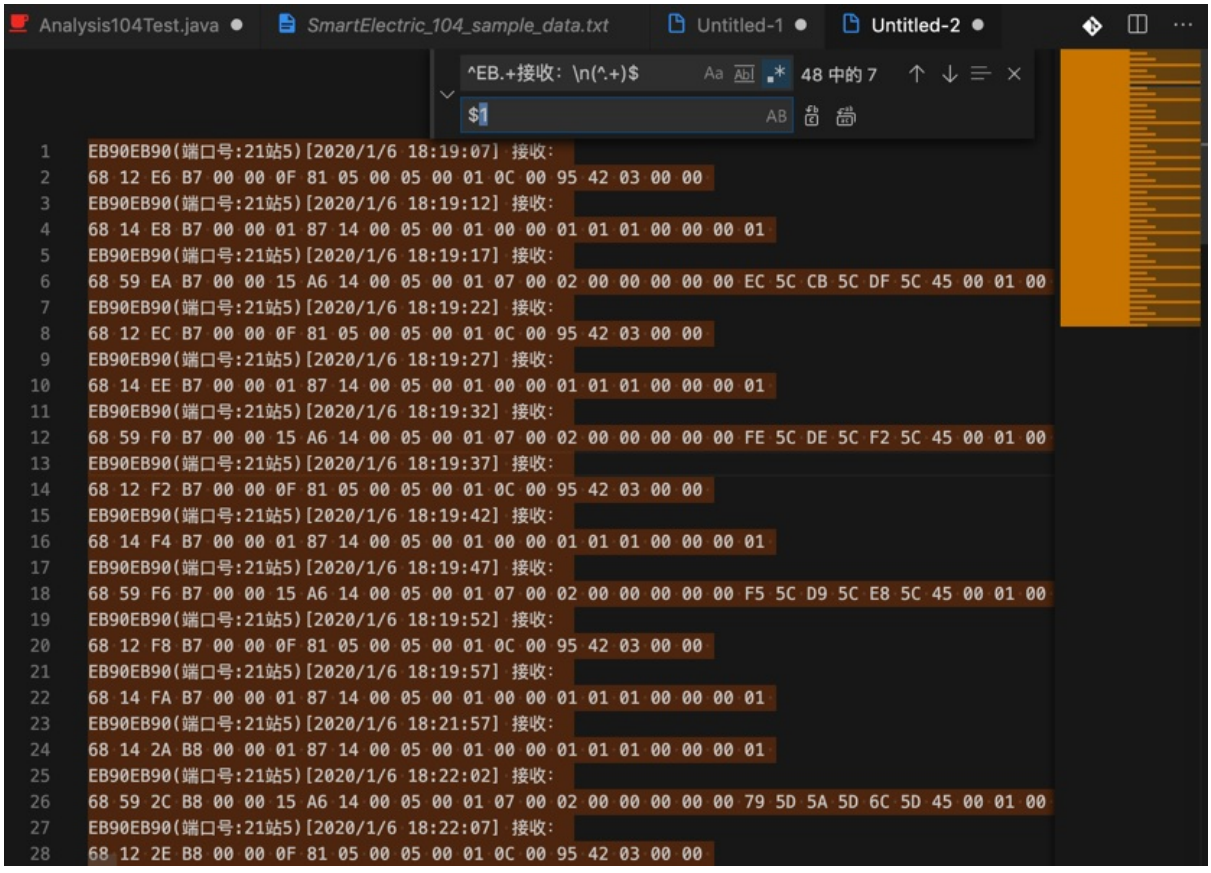
从：

```
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：



替换成：

得到每一行的数字：

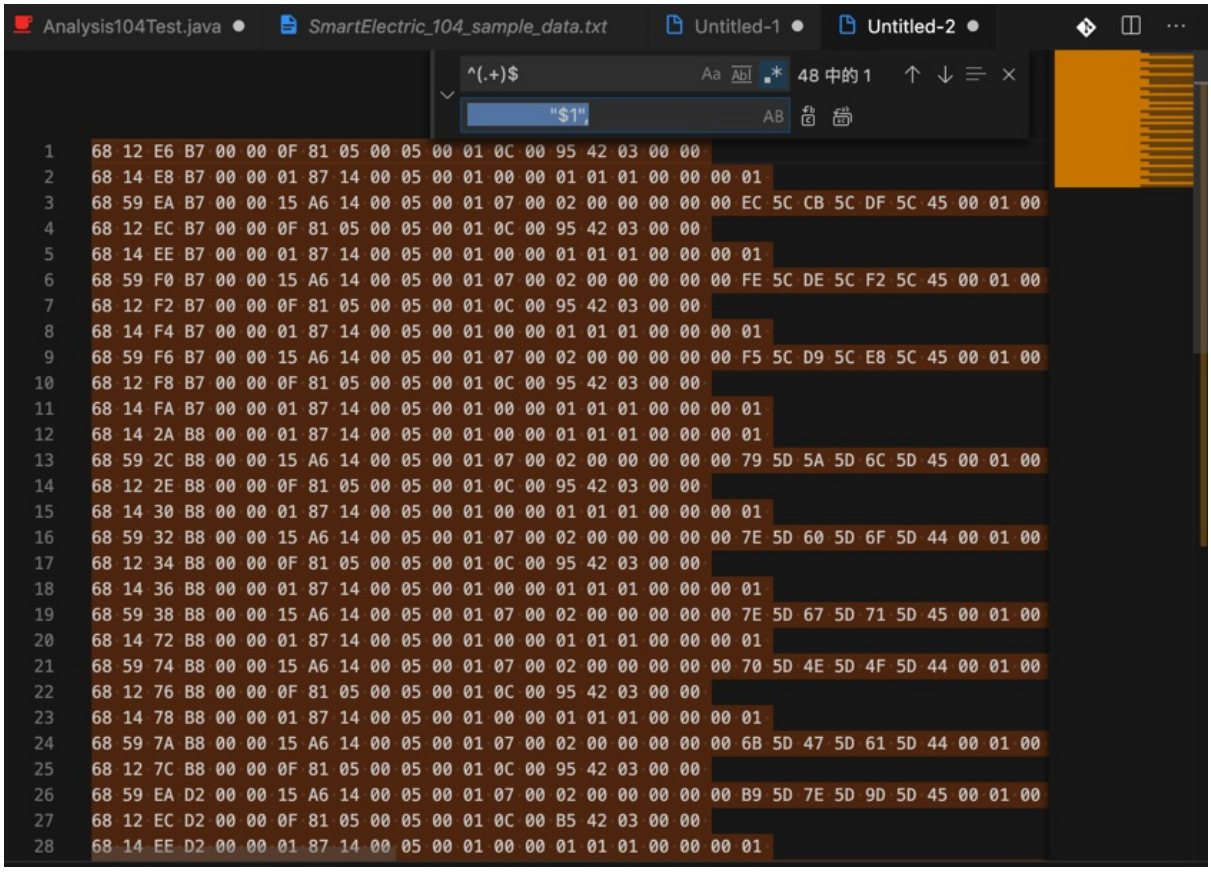
```
68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5C DF 5C 45 00 01
00 FE FF 7C 00 94 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00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

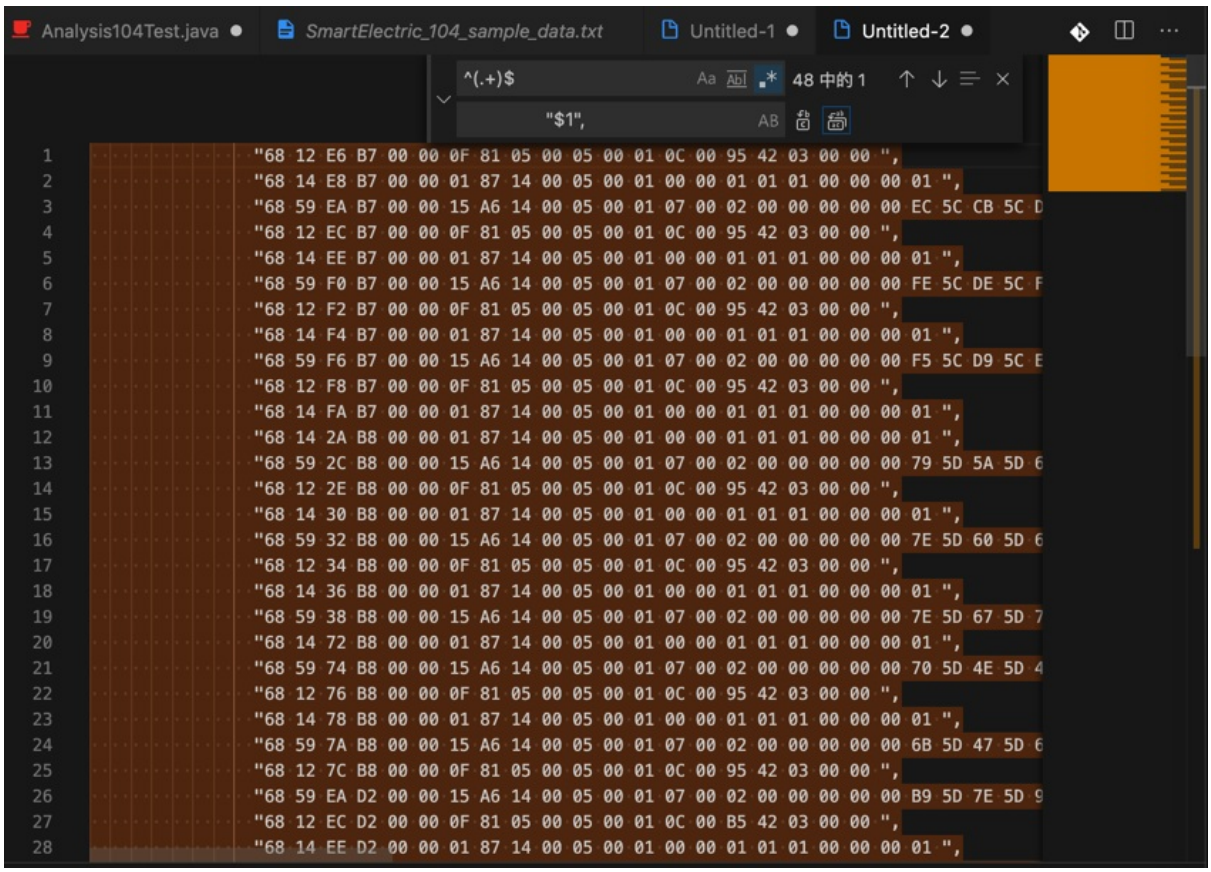
用正则：

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

把：

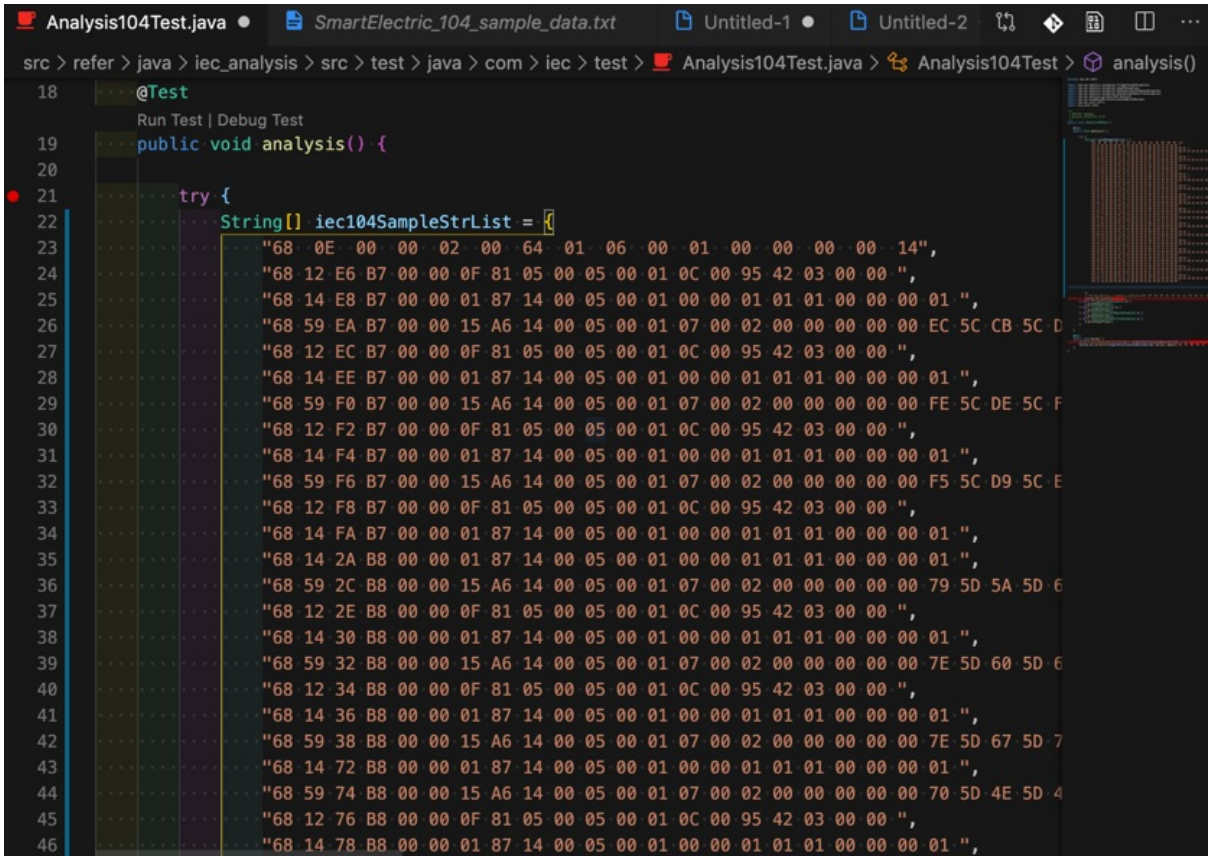

Analysis104Test.java
SmartElectric_104_sample_data.txt
Untitled-1
Untitled-2
^(.+)$
48 中的 1
"$1",
68 12 E6 B7 00 00 0F 81 05 00 05 00 01 0C 00 95 42 03 00 00
68 14 E8 B7 00 00 01 87 14 00 05 00 01 00 00 01 01 01 00 00 00 01
68 59 EA B7 00 00 15 A6 14 00 05 00 01 07 00 02 00 00 00 00 00 EC 5C CB 5C DF 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

变成：


Analysis104Test.java
SmartElectric_104_sample_data.txt
Untitled-1
Untitled-2
^(.+)$
48 中的 1
"$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 D
"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 F
"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 E
"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 6
"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 6
"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 7
"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 4
"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 6
"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 9
"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 ",

```
                "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 00 86 13 00 00 02 00 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：

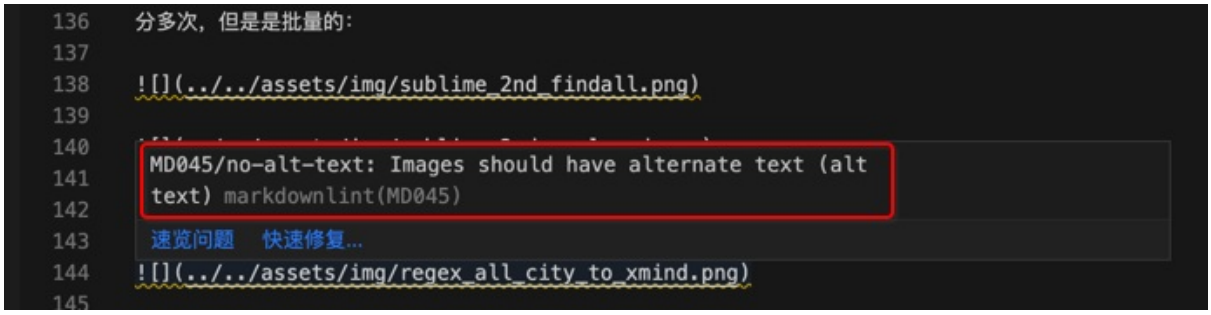


->从而把：

- 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：

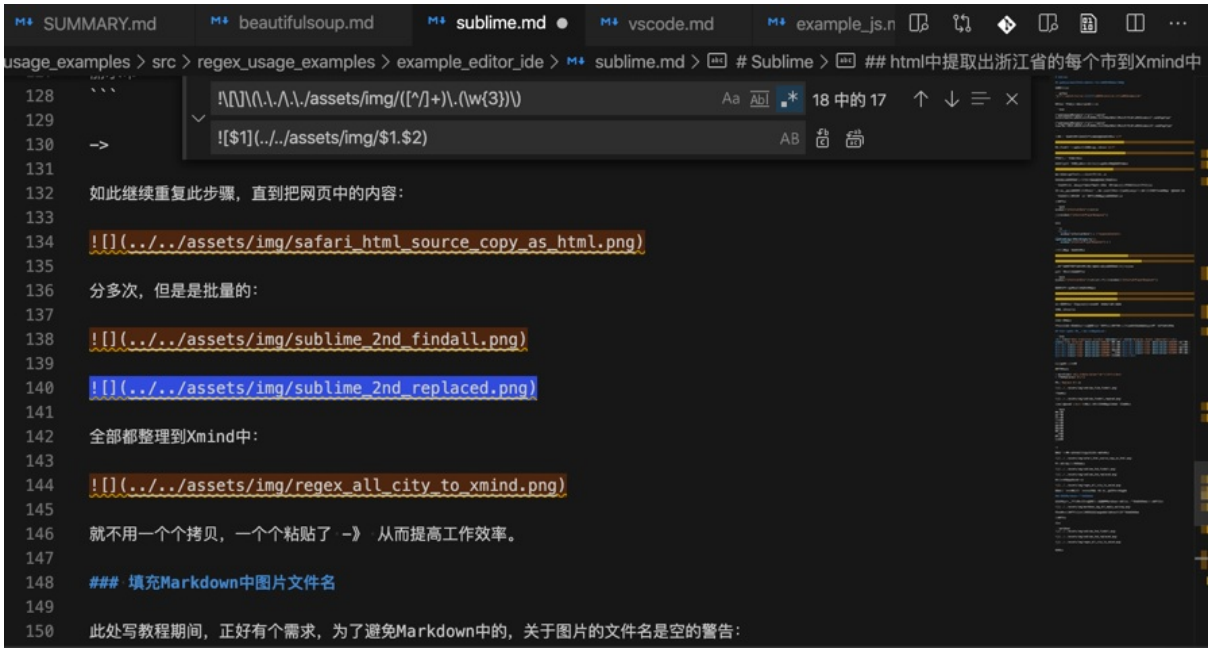
```
\[\]\(\.\./\.\./assets/img/([^/]+)\.(\w{3})\)
[$1](../../assets/img/$1.$2)
```

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)
```

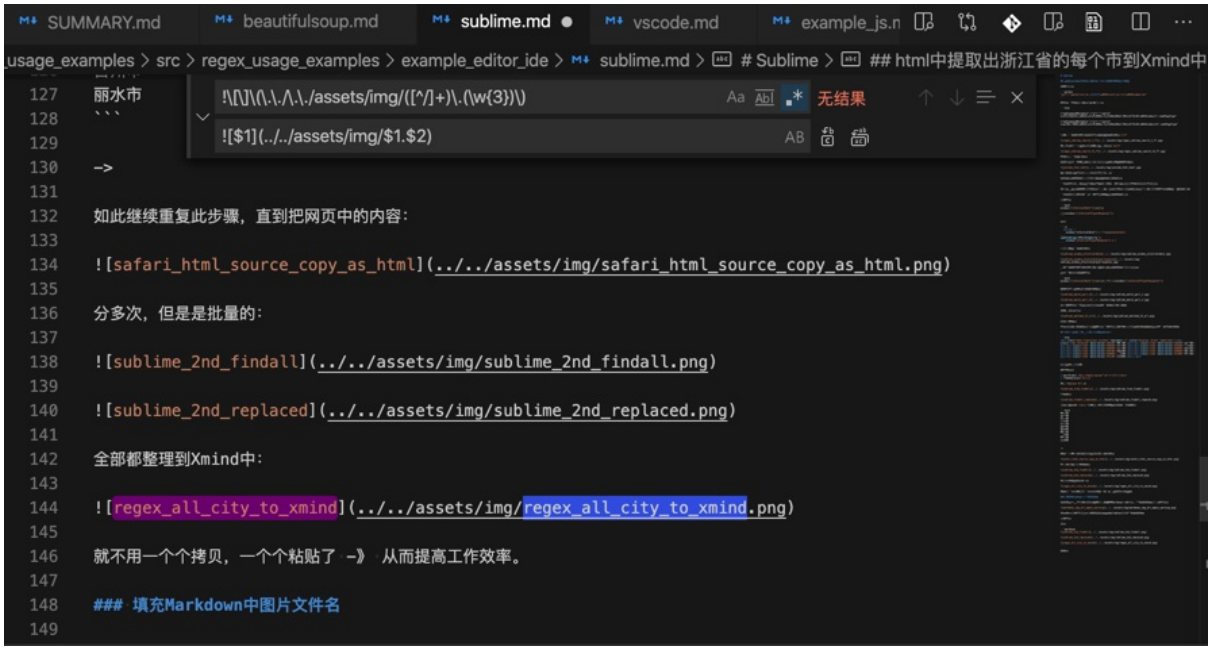


变成：

```
[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。

把xml中非法的大于号和小于号替换掉

用正则批量替换：

```
Callback<([\w\.]+)>
Callback lt;$1gt;
```

从：

```
  <item name="android.accounts.AccountManager android.accounts.AccountManagerFuture&lt;and
roid.os.Bundle&gt; removeAccount(android.accounts.Account, android.app.Activity, android.a
ccounts.AccountManagerCallback<android.os.Bundle>, android.os.Handler)">
...
  <item name="android.accounts.AccountManager android.accounts.AccountManagerFuture&lt;and
roid.os.Bundle&gt; updateCredentials(android.accounts.Account, java.lang.String, android.o
s.Bundle, android.app.Activity, android.accounts.AccountManagerCallback<android.os.Bundle>
, android.os.Handler)”>
...
```

替换成：

```
  <item name="android.accounts.AccountManager android.accounts.AccountManagerFuture&lt;and
roid.os.Bundle&gt; removeAccount(android.accounts.Account, android.app.Activity, android.a
ccounts.AccountManagerCallback&lt;android.os.Bundle&gt;, android.os.Handler)”>
...
  <item name="android.accounts.AccountManager android.accounts.AccountManagerFuture&lt;and
roid.os.Bundle&gt; updateCredentials(android.accounts.Account, java.lang.String, android.o
s.Bundle, android.app.Activity, android.accounts.AccountManagerCallback&lt;android.os.Bund
le&gt;, android.os.Handler)">
```

以及更加复杂一点的：

`Android/sdk/platform-tools/api/android/hardware/camera2/annotations.xml`

```
  <item name="android.hardware.camera2.CameraDevice android.hardware.camera2.CaptureReques
t.Builder createCaptureRequest(int, java.util.Set<java.lang.String>) 0">
    <annotation name="androidx.annotation.IntDef">
      <val name="value" val="{android.hardware.camera2.CameraDevice.TEMPLATE_PREVIEW, andr
oid.hardware.camera2.CameraDevice.TEMPLATE_STILL_CAPTURE, android.hardware.camera2.CameraD
evice.TEMPLATE_RECORD, android.hardware.camera2.CameraDevice.TEMPLATE_VIDEO_SNAPSHOT, andr
oid.hardware.camera2.CameraDevice.TEMPLATE_ZERO_SHUTTER_LAG, android.hardware.camera2.Came
raDevice.TEMPLATE_MANUAL}" />
    </annotation>
  </item>
```

要把其中的：

```
java.util.Set<java.lang.String>
```

替换成：

```
java.util.Set&lt;java.lang.String&gt;
```

且，又不想

- 很傻的把前缀写成固定的：`java.util.Set`
  - 这样前缀换了，就不支持了
    - 虽然此处只有这一个例子，但是还是应该写的更加通用
- 以及后续打算放到全局去替换的
  - 所以要写成通用的

参考自己教程

环视断言 · 应用广泛的超强搜索：正则表达式

```
(?<xxx): (positive) look ahead (assertion) 正向肯定断言
(?<xxx): negative look ahead (assertion) 正向否定断言
(?<=xxx): (positive) look behind (assertion) 反向肯定断言
(?<!xxx): negative look behind (assertion) 反向否定断言
```

然后去写成：

```
(?<=\s)<([\.\w]+)>
&lt;$1&gt;
```

即可：只识别中间内容，不会误判

可以把：

```
  <item name="android.hardware.camera2.CameraDevice android.hardware.camera2.CaptureReques
t.Builder createCaptureRequest(int, java.util.Set<java.lang.String>) 0">
```

替换成：

```
  <item name="android.hardware.camera2.CameraDevice android.hardware.camera2.CaptureReques
t.Builder createCaptureRequest(int, java.util.Set&lt;java.lang.String&gt;) 0">
```

把表格文字用正则处理成markdown表格，用于生成html后，拷贝出真正表格

从某网站 Android应用加固原理 - 简书，希望拷贝出 表格，但是只是拷贝出文字：

```
风险名称	风险	解决方案
1.App防止反编译	被反编译的暴露客户端逻辑，加密算法，密钥，等等	加固
2.java层代码源代码反编译风险	被反编译的暴露客户端逻辑，加密算法，密钥，等等	加固 ，混淆
3.so文件破解风险	导致核心代码泄漏。	so文件加固
4.篡改和二次打包风险	修改文件资源等，二次打包的添加病毒，广告，或者窃取支付密码，拦截短信等	资源文件混淆和校验签名的hash值
5.资源文件泄露风险	获取图片，js文件等文件，通过植入病毒，钓鱼页面获取用户敏感信息	资源混淆，加固等等
6.应用签名未交验风险	反编译或者二次打包，添加病毒代码，恶意代码，上传盗版App	对App进行签名证书校验
7.代码为混淆风险	业务逻辑暴露，加密算法，账号信息等等。	混淆（中文混淆）
8.webview明文存储密码风险	用户使用webview默认存储密码到databases/webview.db root的手机可以产看webview数据库，获取用户敏感信息	关闭wenview存储密码功能
9.明文数字证书风险	APK使用的数字证书用来校验服务器的合法性，保证数据的保密性和完成性 明文存储的证书被篡改造成数据被获取等	客户端校验服务器域名和数字证书等
10.调试日志函数调用风险	日志信息里面含有用户敏感信息等	关闭调试日志函数，删除打印的日志信息
11.AES/DES加密方法不安全使用风险	在使用AES/DES加密使用了ECB或者OFB工作模式，加密数据被选择明文攻击破解等	使用CBC和CFB工作模式等
12.RSA加密算法不安全风险	密数据被选择明文攻击破解和中间人攻击等导致用户敏感信息泄露	密码不要太短，使用正确的工作模式
13.密钥硬编码风险	用户使用加密算法的密钥设置成一个固定值导致密钥泄漏	动态生成加密密钥或者将密钥进程分段存储等
14.动态调试攻击风险	攻击者使用GDB，IDA调试追踪目标程序，获取用户敏感信息等	在so文件里面实现对调试进程的监听
15.应用数据任意备份风险	AndroidMainfest中allowBackup true 攻击者可以使用adb命令对APP应用数据进行备份造成用户数据泄露	allowBackup false
16.全局可读写内部文件风险。	实现不同软件之间数据共享，设置内部文件全局可读写造成其他应用也可以读取或者修改文件等	（1）.使用MODE_PRIVATE模式创建内部存储文件（2）.加密存储敏感数据3.避免在文件中存储明文和敏感信息
17.SharedPrefs全局可读写内部文件风险。	被其他应用读取或者修改文件等	使用正确的权限
18.Internal Storage数据全局可读写风险	当设置MODE_WORLD_READBLE或者设置android:sharedUserId导致敏感信息被其他应用程序读取等	设置正确的模式等
19.getDir数据全局可读写风险	当设置MODE_WORLD_READBLE或者设置android:sharedUserId导致敏感信息被其他应用程序读取等	设置正确的模式等
20.java层动态调试风险	AndroidManifest中调试的标记可以使用jdb进行调试，窃取用户敏感信息。	a
```

```
ndroid: debuggable “false”
21.内网测试信息残留风险　　通过测试的Url，测试账号等对正式服务器进行攻击等　　讲测试内网的日志清除，或者测试服务器和生产服务器不要使用同一个
22.随机数不安全使用风险　　在使用SecureRandom类来生成随机数，其实并不是随机，导致使用的随机数和加密算法被破解。　　（1）不使用setSeed方法（2）使用/dev/urandom或者/dev/random来初始化伪随机数生成器
23.Http传输数据风险　　未加密的数据被第三方获取，造成数据泄露　　使用Hpps
24.Htpps未校验服务器证书风险，Https未校验主机名风险，Https允许任意主机名风险　　客户端没有对服务器进行身份完整性校验，造成中间人攻击　　（1）.在X509TrustManager中的checkServerTrusted方法对服务器进行校验（2）.判断证书是否过期（3）.使用HostnameVerifier类检查证书中的主机名与使用证书的主机名是否一致
25.webview绕过证书校验风险　　webview使用https协议加密的url没有校验服务器导致中间人攻击　　校验服务器证书时候正确
26.界面劫持风险　　用户输入密码的时候被一个假冒的页面遮挡获取用户信息等　　（1）.使用第三方专业防界面劫持SDK（2）.校验当前是否是自己的页面
27.输入监听风险　　用户输入的信息被监听或者按键位置被监听造成用户信息泄露等　　自定义键盘
28.截屏攻击风险　　对APP运行中的界面进行截图或者录制来获取用户信息　　添加属性getWindow().setFlags(FLAG_SECURE)不让用户截图和录屏
29.动态注册Receiver风险　　当动态注册Receiver默认生命周期是可以导出的可以被任意应用访问　　使用带权限检验的registerReceiver API进行动态广播的注册
30.Content Provider数据泄露风险　　权限设置不当导致用户信息　　正确的使用权限
31.Service ，Activity，Broadcast,content provider组件导出风险　　Activity被第三方应用访问导致被任意应用恶意调用　　自定义权限
32.PendingIntent错误使用Intent风险　　使用PendingIntent的时候，如果使用了一个空Intent，会导致恶意用户劫持修改Intent的内容　　禁止使用一个空Intent去构造PendingIntent
33.Intent组件隐式调用风险　　使用隐式Intent没有对接收端进行限制导致敏感信息被劫持　　1.对接收端进行限制 2.建议使用显示调用方式发送Intent
34.Intent Scheme URL攻击风险　　webview恶意调用App　　对Intent做安全限制
35.Fragment注入攻击风险　　出的PreferenceActivity的子类中，没有加入isValidFragment方法，进行fragment名的合法性校验，攻击者可能会绕过限制，访问未授权的界面　　（1）.如果应用的Activity组件不必要导出，或者组件配置了intent filter标签，建议显示设置组件的“android:exported”属性为false（2）.重写isValidFragment方法，验证fragment来源的正确性
36.webview远程代码执行风险　　风险：WebView.addJavascriptInterface方法注册可供JavaScript调用的Java对象，通过反射调用其他java类等　　建议不使用addJavascriptInterface接口，对于Android API Level为17或者以上的Android系统，Google规定允许被调用的函数，必须在Java的远程方法上面声明一个@JavascriptInterface注解
37.zip文件解压目录遍历风险　　Java代码在解压ZIP文件时，会使用到ZipEntry类的getName()方法，如果ZIP文件中包含“../”的字符串，该方法返回值里面原样返回，如果没有过滤掉getName()返回值中的“../”字符串，继续解压缩操作，就会在其他目录中创建解压的文件　　（1）．对重要的ZIP压缩包文件进行数字签名校验，校验通过才进行解压。（2）．检查Zip压缩包中使用ZipEntry.getName()获取的文件名中是否包含”../”或者”../”，检查”../”的时候不必进行URI Decode（以防通过URI编码”..%2F”来进行绕过），测试发现ZipEntry.getName()对于Zip包中有“..%2F”的文件路径不会进行处理。
38.Root设备运行风险　　已经root的手机通过获取应用的敏感信息等　　检测是否是root的手机禁止应用启动
39.模拟器运行风险　　刷单，模拟虚拟位置等　　禁止在虚拟器上运行
40.从sdcard加载Dex和so风险　　未对Dex和So文件进行安全，完整性及校验，导致被替换，造成用户敏感信息泄露　　（1）.放在APP的私有目录 （2）.对文件进行完成性校验。
```

然后希望处理成markdown

用正则替换：

```
(.+?)\t(.+?)\t(.+?)$
```

```
$1 $2 $3
```

后记： 此处 `\t` 的 `Tab` 粘贴过来已变成 `空格` = `space` 了，所以改为：

```
(.+?)\s+(.+?)\s+(.+?)$
$1 $2 $3
```

效果：

从：

【此处由于印象笔记冲突导致帖子图片丢失】

变成：

【此处由于印象笔记冲突导致帖子图片丢失】

保存成 md文件，再去用VSCode中Markdown插件生成html或pdf

不过要加上一个表格的内容

```
----- ---- -----
```

打开效果：

【此处由于印象笔记冲突导致帖子图片丢失】

然后也方便，拷贝到别处，是个表格：

比如拷贝到印象笔记中：

【此处由于印象笔记冲突导致帖子图片丢失】

把url中query parameter分出来

折腾：

【未解决】爬取mp.codeup.cn中的英语教材电子书资源

期间，对于：

```
http://mp.codeup.cn/book/sample2.htm?id 52365 shelfId 4824 share_ 6765370 sh sh vt_ 158311
1113754 _logined 1
```

想要把参数分出来

用正则：

```
[^?](.+?)=([^&]+)
\n$1 = $2
```

从：

【此处由于印象笔记冲突导致帖子图片丢失】

变成：

```
http://mp.codeup.cn/book/sample2.htm
id = 52365
shelfId = 4824
share_ = 6765370
sh = sh
vt_ = 1583111113754
_logined = 1
```

除了第一行，剩下的就是我们要的 `key=value` 形式了。

其他小的例子

此处整理其他相对小的零碎的例子供参考。

把print换成logging.debug

用：

```
print\("([^"]+)"(\s*%\s*\(?([^\)]+)\)?)?\)
logging.debug("$1", $3)
```

把：

print没有参数的：

```
print("taped 通讯录")
```

print单个参数的

```
print("++++++++++ taped element: %s" % curElement)
```

print多个参数的：

```
print("++++++++++ clicked element position: %s,%s" % (centerX, centerY))
print("Cost time %.2fs for save source %s" % (saveSourceTime, savedSourceFile))
```

都一次性变成logging.debug的写法：

```
      logging.debug("++++++++++ taped element: %s", curElement)

      logging.debug("++++++++++ clicked element position: %s,%s", centerX, centerY)

    logging.debug("Cost time %.2fs for save source %s", saveSourceTime, savedSourceFile)

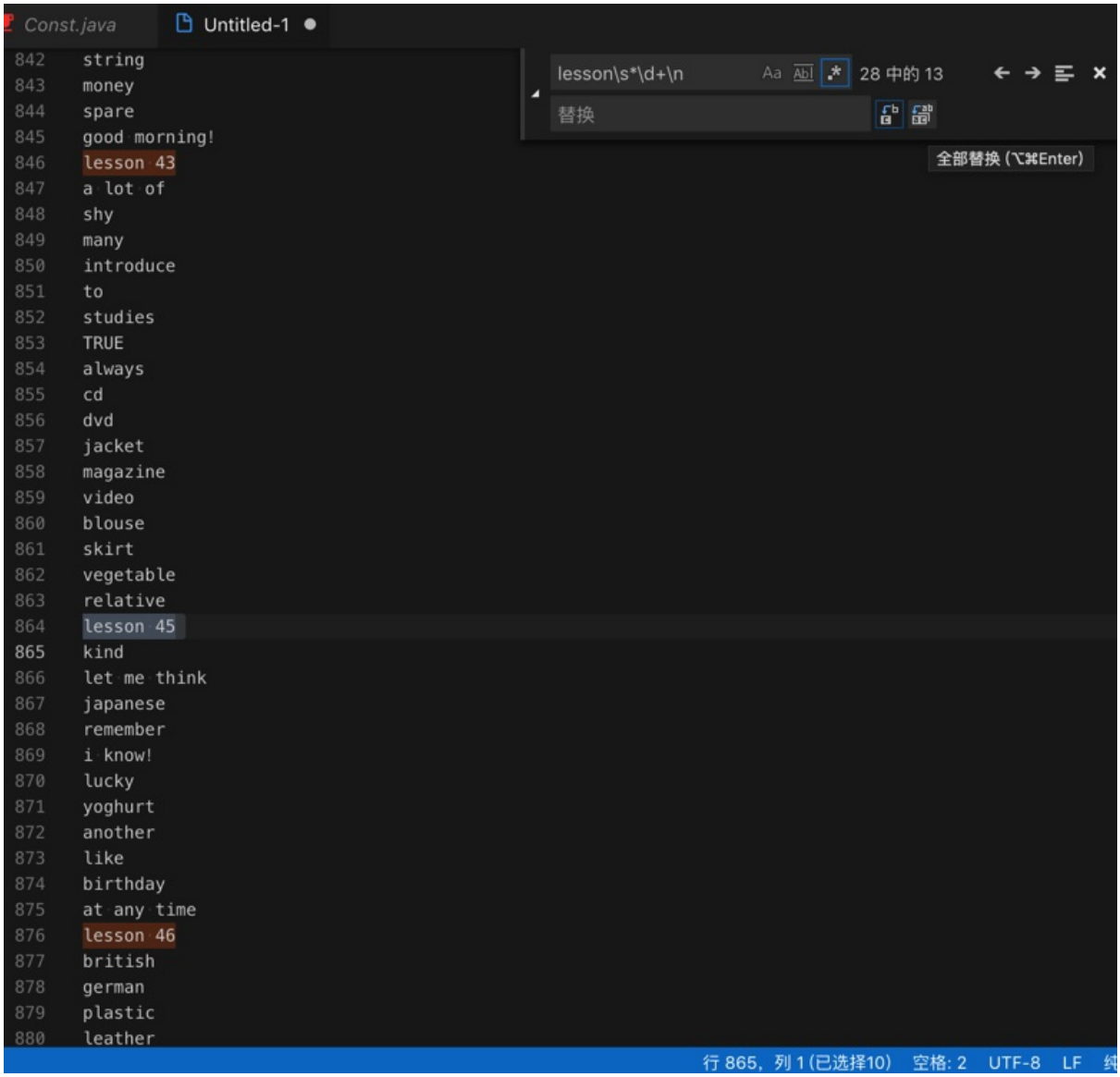
    logging.debug("taped 通讯录", )
```

去除内容中多余的 lessionxxx的单词

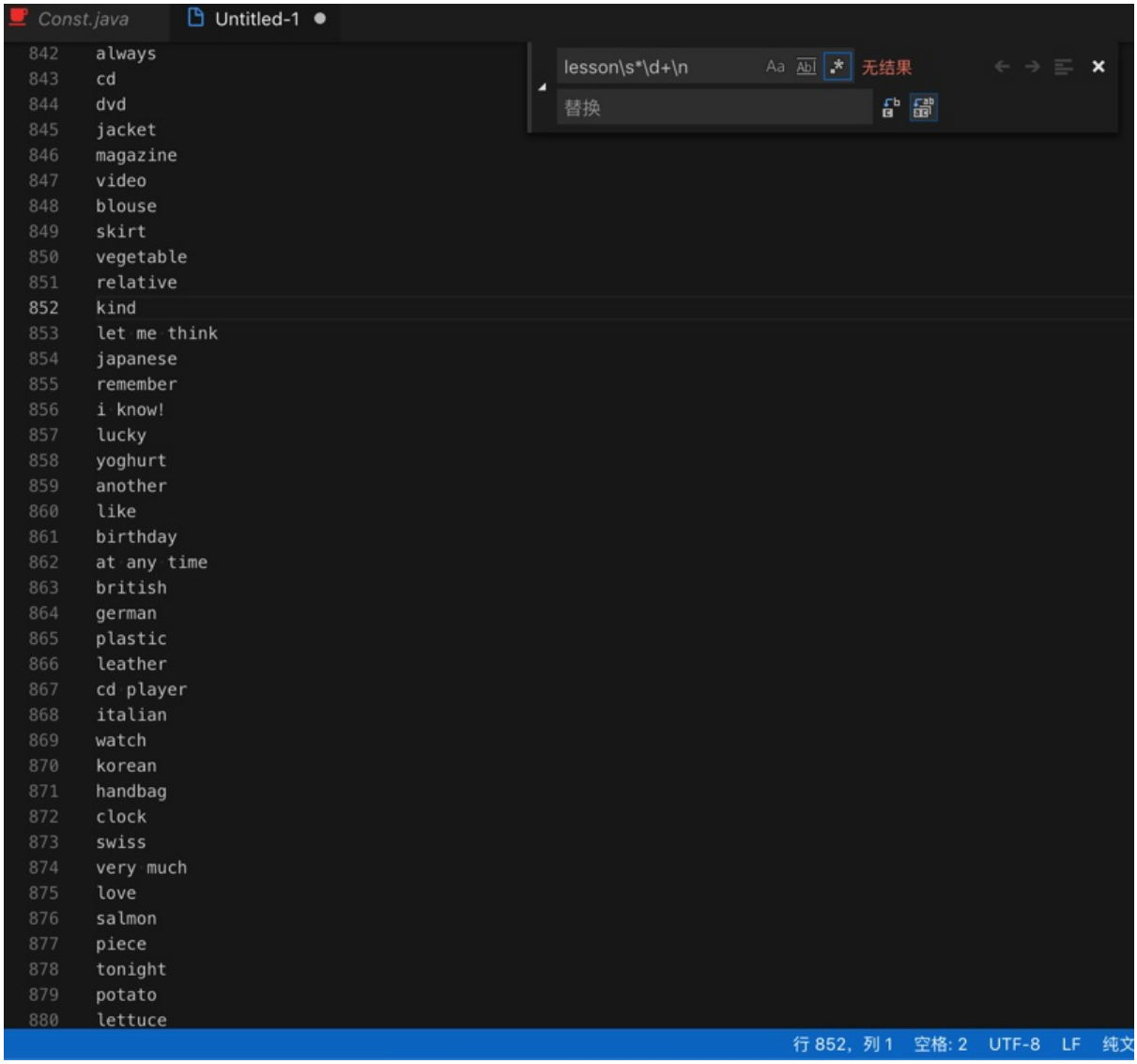
正则：

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

从：


Const.java
Untitled-1
842 string
843 money
844 spare
845 good morning!
846 lesson 43
847 a lot of
848 shy
849 many
850 introduce
851 to
852 studies
853 TRUE
854 always
855 cd
856 dvd
857 jacket
858 magazine
859 video
860 blouse
861 skirt
862 vegetable
863 relative
864 lesson 45
865 kind
866 let me think
867 japanese
868 remember
869 i know!
870 lucky
871 yoghurt
872 another
873 like
874 birthday
875 at any time
876 lesson 46
877 british
878 german
879 plastic
880 leather
lesson\s*\d+\n
28 中的 13
替换
全部替换 (⌥⌘Enter)
行 865，列 1 (已选择10)
空格: 2
UTF-8
LF

替换成：

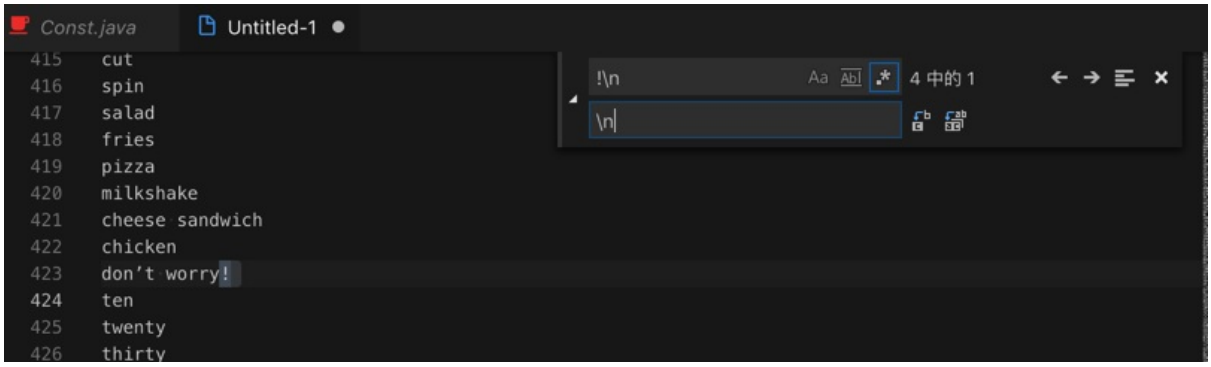


语句末尾去掉感叹号

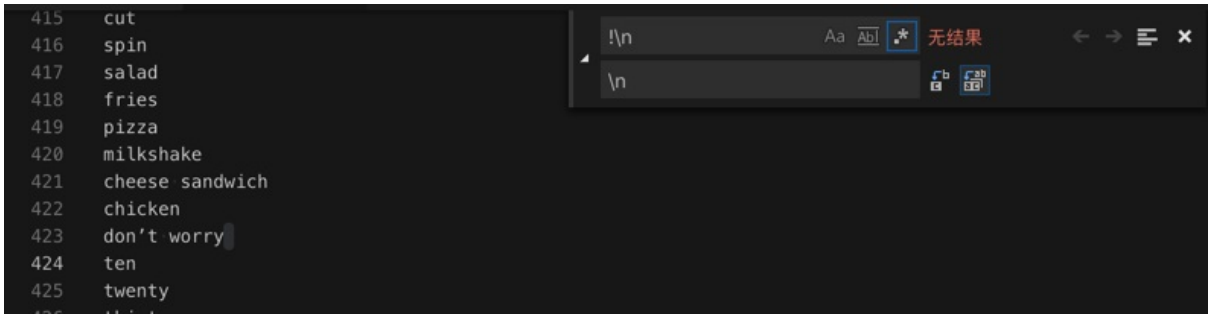
正则：

```
 \n
\n
```

从：



替换成：



文章标题和链接转换为Markdown的链接

正则替换规则：

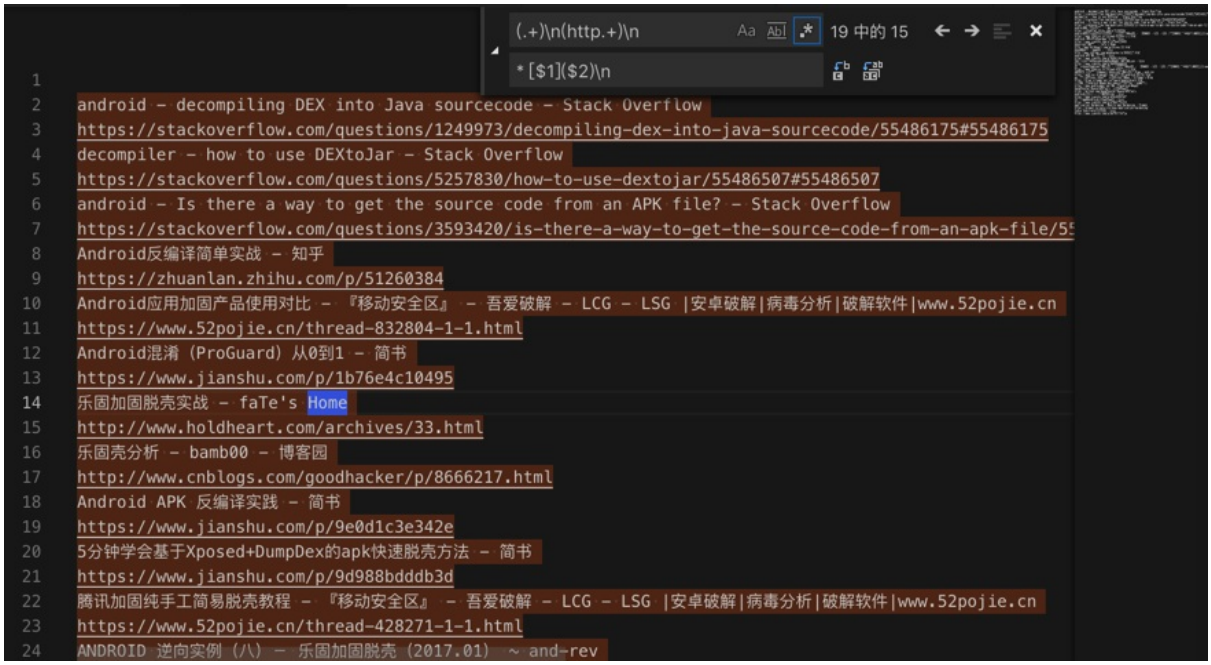
```
(.+)\n(http.+)\n
* [$1]($2)\n
```

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
```

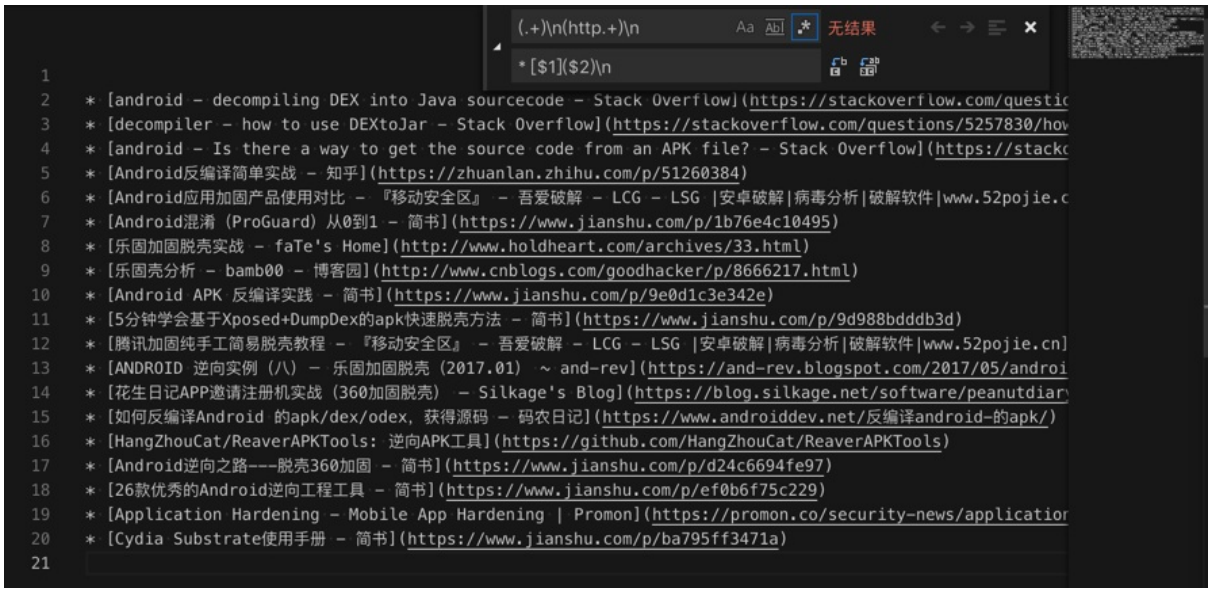
把：



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](htt
```

```
ps://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

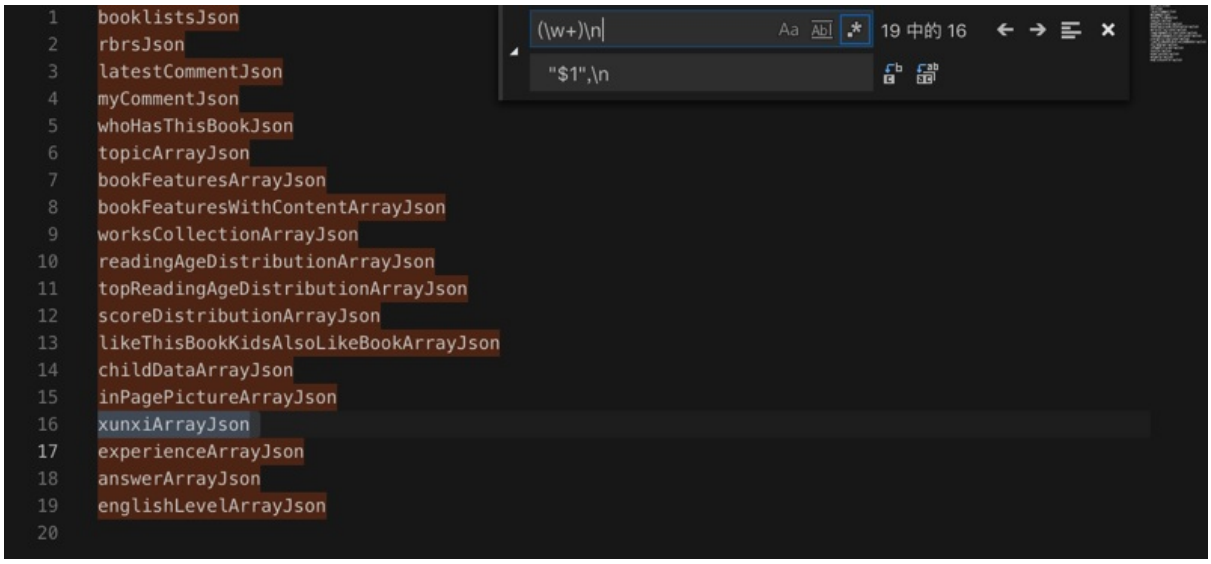
正则：

```
(\w+)\n
```

```
  "$1",\n
```

从：

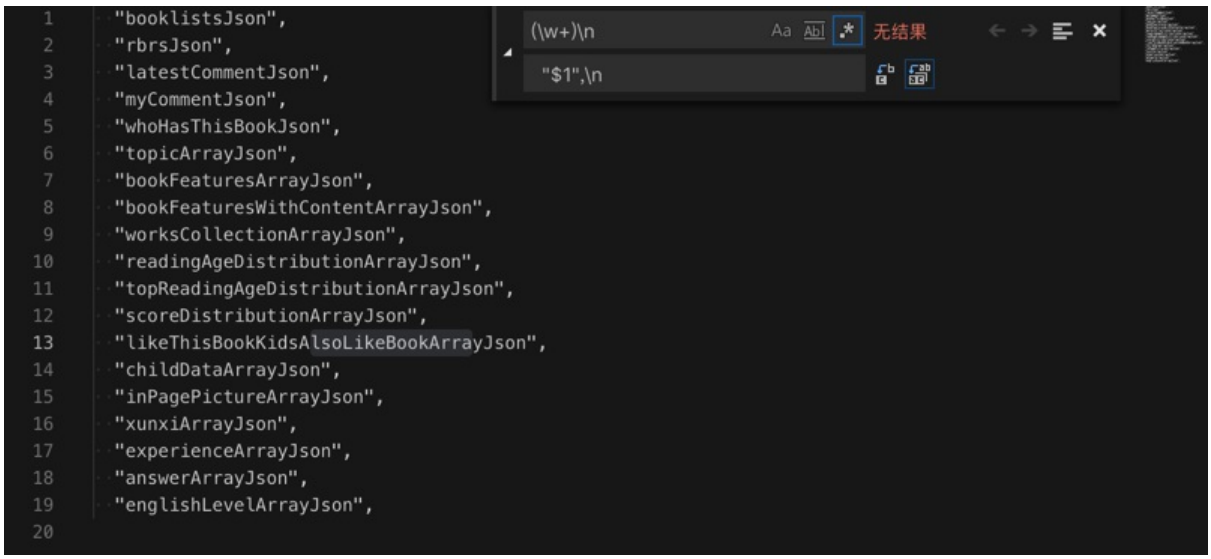
```
booklistsJson
rbrsJson
latestCommentJson
myCommentJson
whoHasThisBookJson
topicArrayJson
bookFeaturesArrayJson
bookFeaturesWithContentArrayJson
worksCollectionArrayJson
readingAgeDistributionArrayJson
topReadingAgeDistributionArrayJson
scoreDistributionArrayJson
likeThisBookKidsAlsoLikeBookArrayJson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



变成：

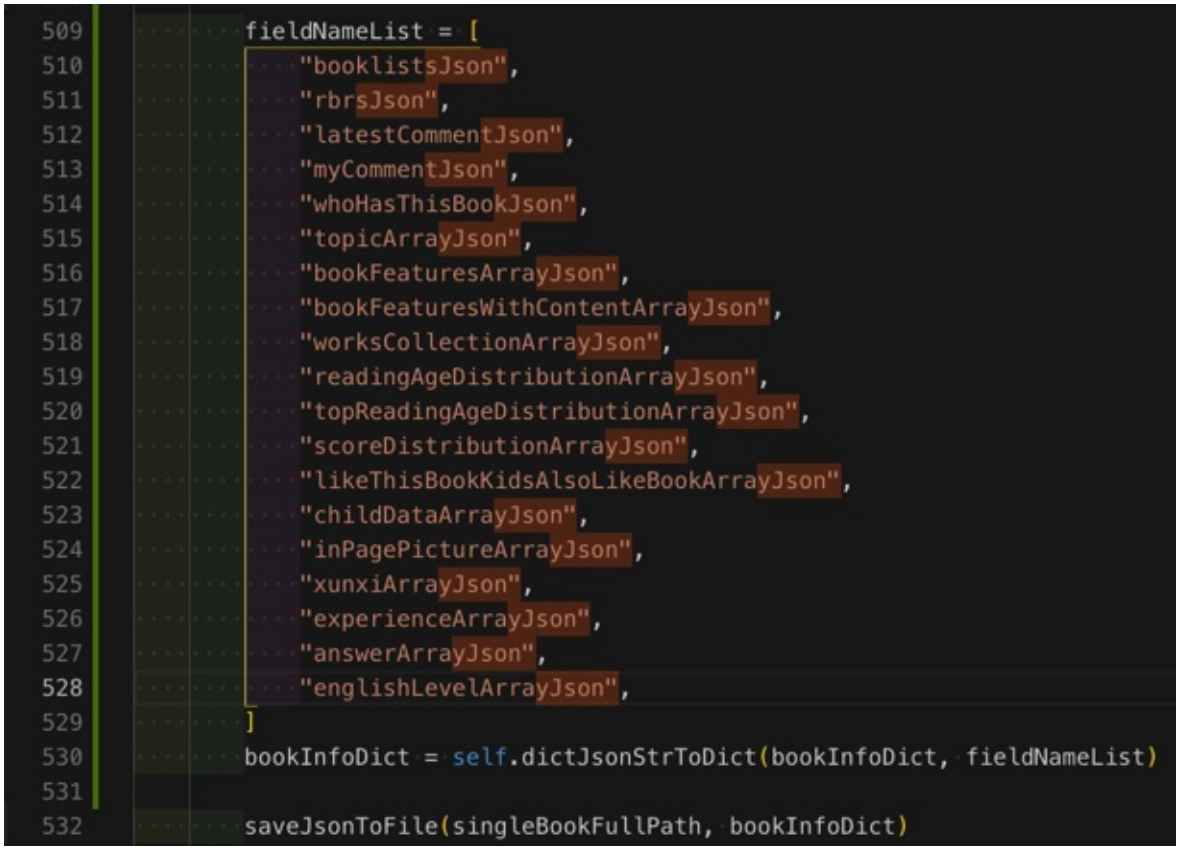
```
  "booklistsJson",
  "rbrsJson",
  "latestCommentJson",
  "myCommentJson",
  "whoHasThisBookJson",
  "topicArrayJson",
  "bookFeaturesArrayJson",
  "bookFeaturesWithContentArrayJson",
  "worksCollectionArrayJson",
```

```
    "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：



```
fieldNameList = [
    "booklistsJson",
    "rbrsJson",
    "latestCommentJson",
    "myCommentJson",
    "whoHasThisBookJson",
    "topicArrayJson",
    "bookFeaturesArrayJson",
    "bookFeaturesWithContentArrayJson",
    "worksCollectionArrayJson",
    "readingAgeDistributionArrayJson",
    "topReadingAgeDistributionArrayJson",
    "scoreDistributionArrayJson",
    "likeThisBookKidsAlsoLikeBookArrayJson",
    "childDataArrayJson",
    "inPagePictureArrayJson",
    "xunxiArrayJson",
    "experienceArrayJson",
    "answerArrayJson",
    "englishLevelArrayJson",
]
bookInfoDict = self.dictJsonStrToDict(bookInfoDict, fieldNameList)

saveJsonToFile(singleBookFullPath, bookInfoDict)
```

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：

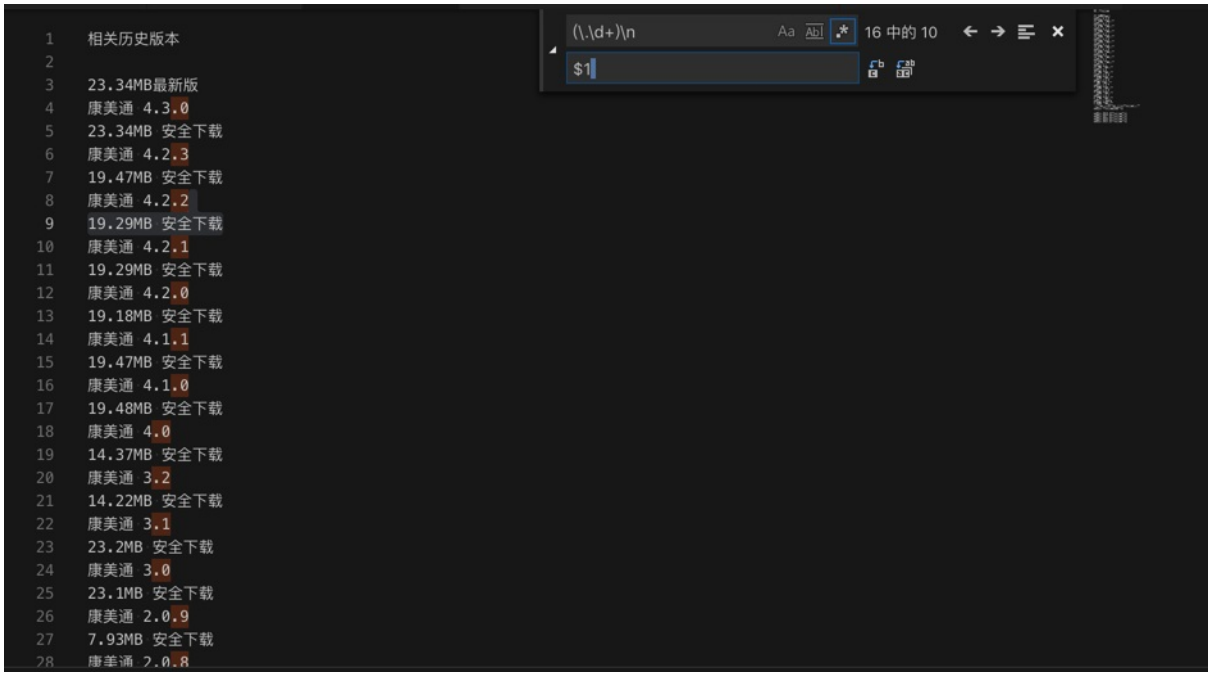
```
(\.\d+)\n
$1
```

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
```



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
```

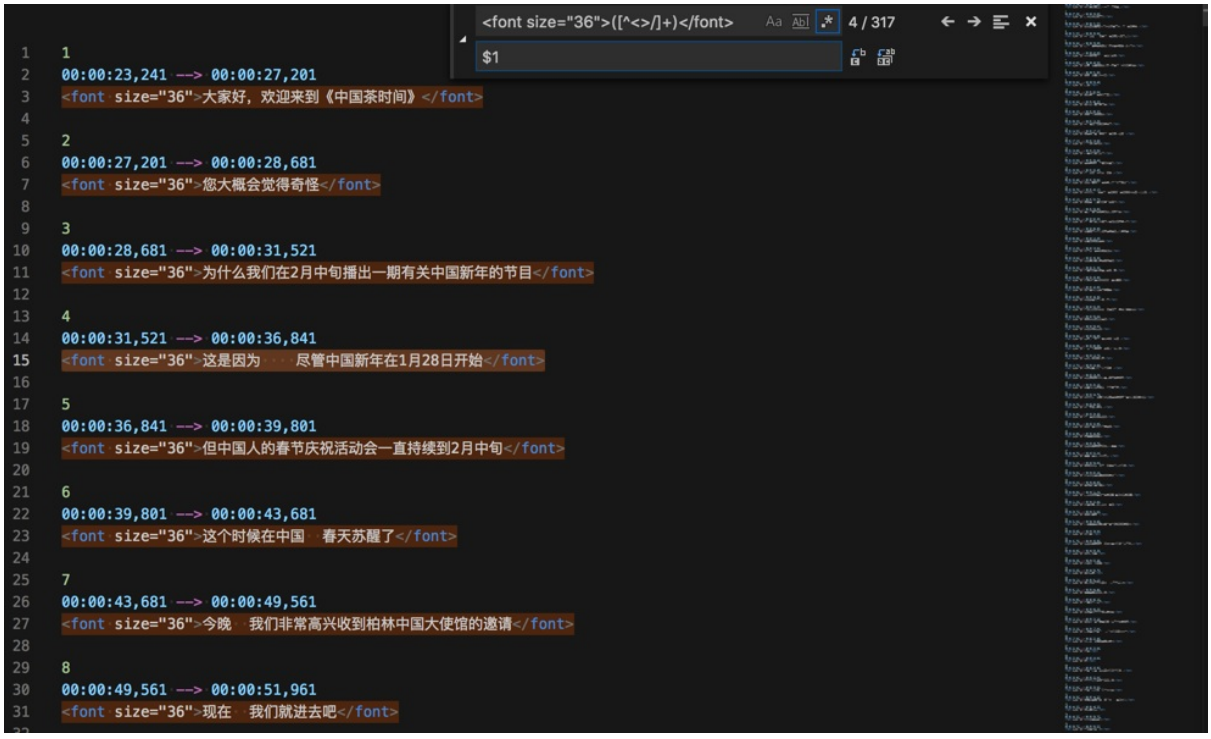
```
康美通 4.2.0 19.18MB
康美通 4.1.1 19.47MB
康美通 4.1.0 19.48MB
康美通 4.0 14.37MB
康美通 3.2 14.22MB
康美通 3.1 23.2MB
康美通 3.0 23.1MB
康美通 2.0.9 7.93MB
康美通 2.0.8 7.3MB
康美通 2.0.7 7.29MB
康美通 2.0.6 7.29MB
康美通 1.0.1 6.27MB
康美通 1.0beta 5.86MB
```

去掉srt字幕中font size

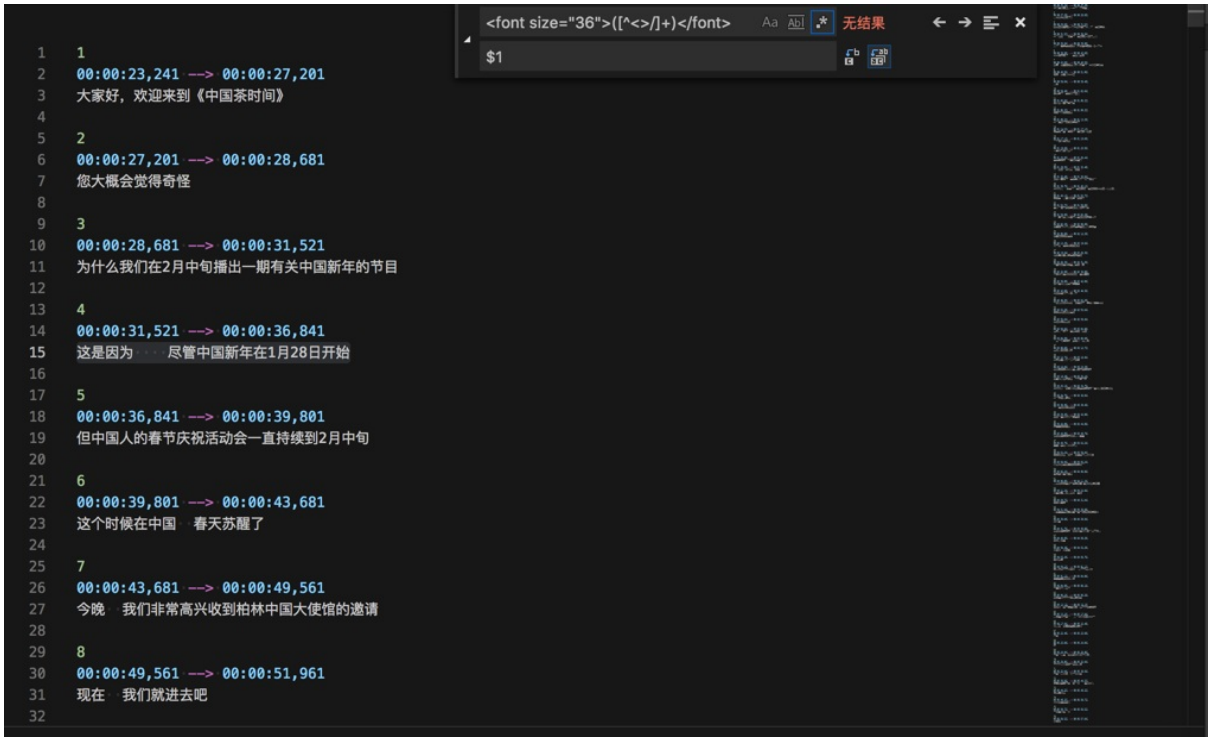
正则：

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

从：



替换成：



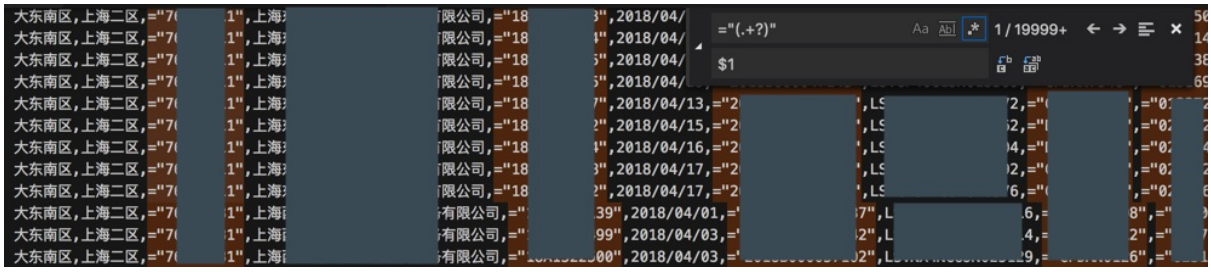
去除掉csv中多余的 `="xxx"`

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。

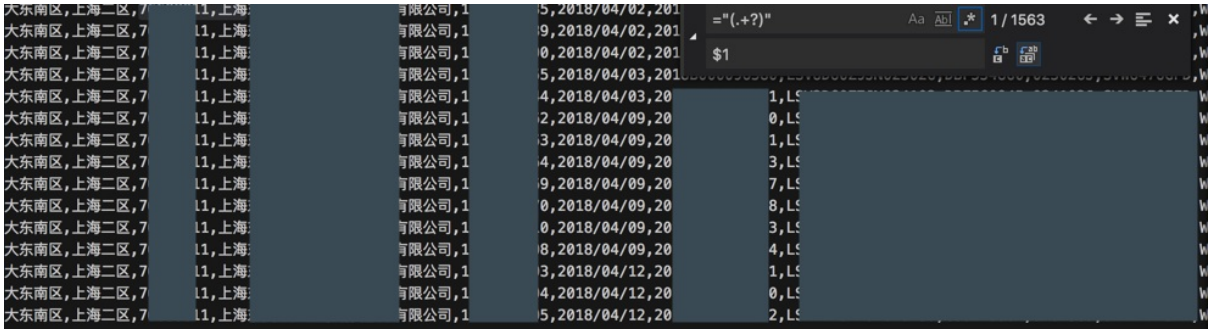
所以用VSCode去替换，用正则：

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

实现了，把 `="7xxx1"`：



替换成 `7xxx1`：



即可。

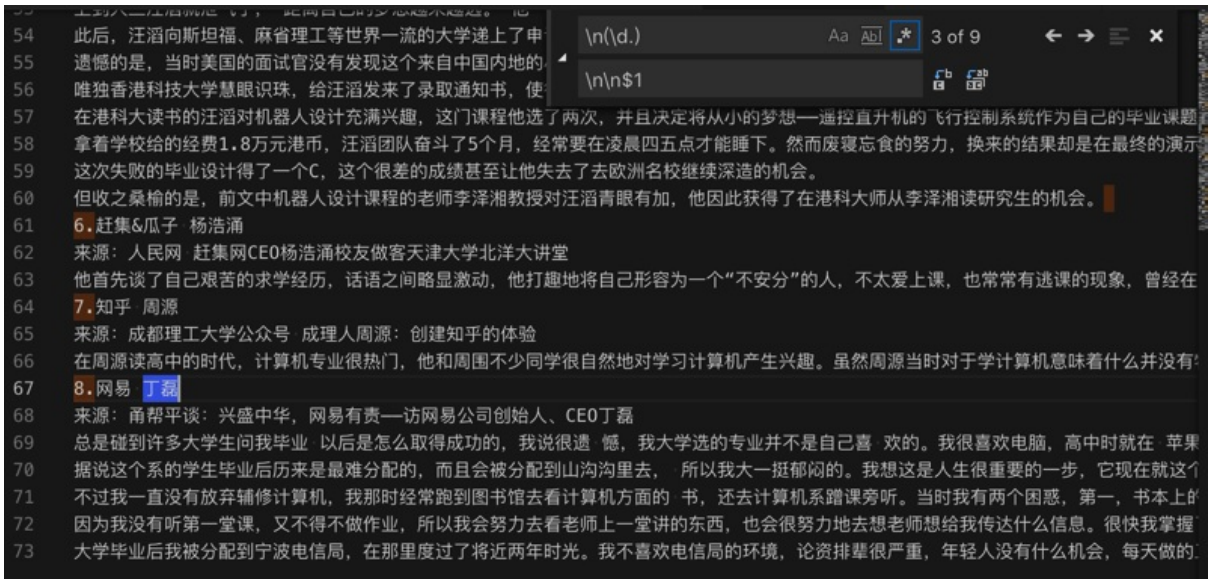
给文章段落增加换行

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
```

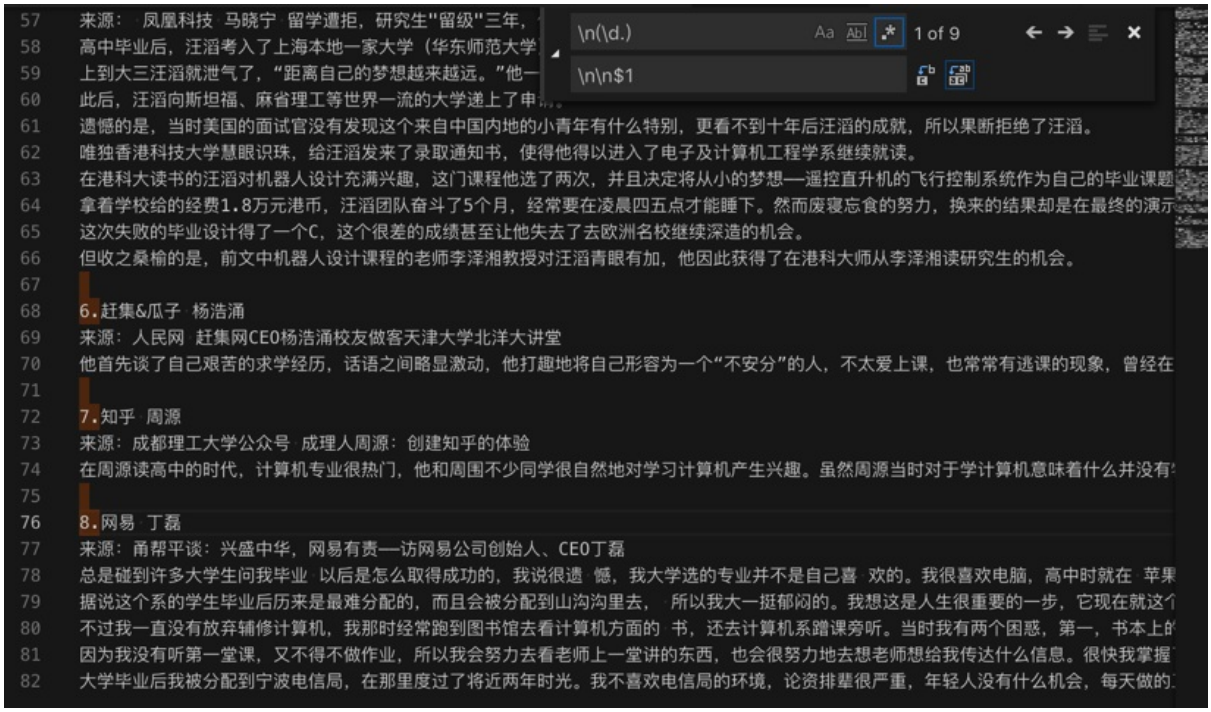
...说过。

2.陆奇给年轻人的话

...

8.网易 丁磊

...



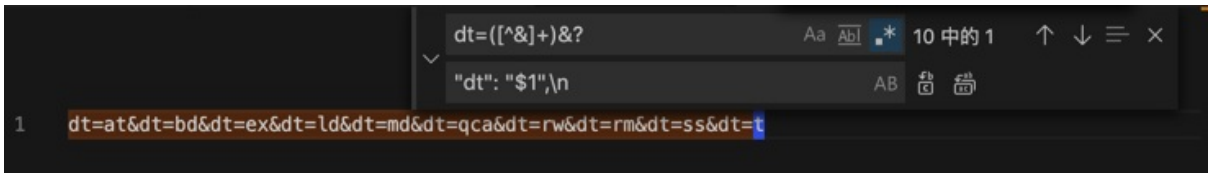
把url中查询参数换成代码中字典参数

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
```

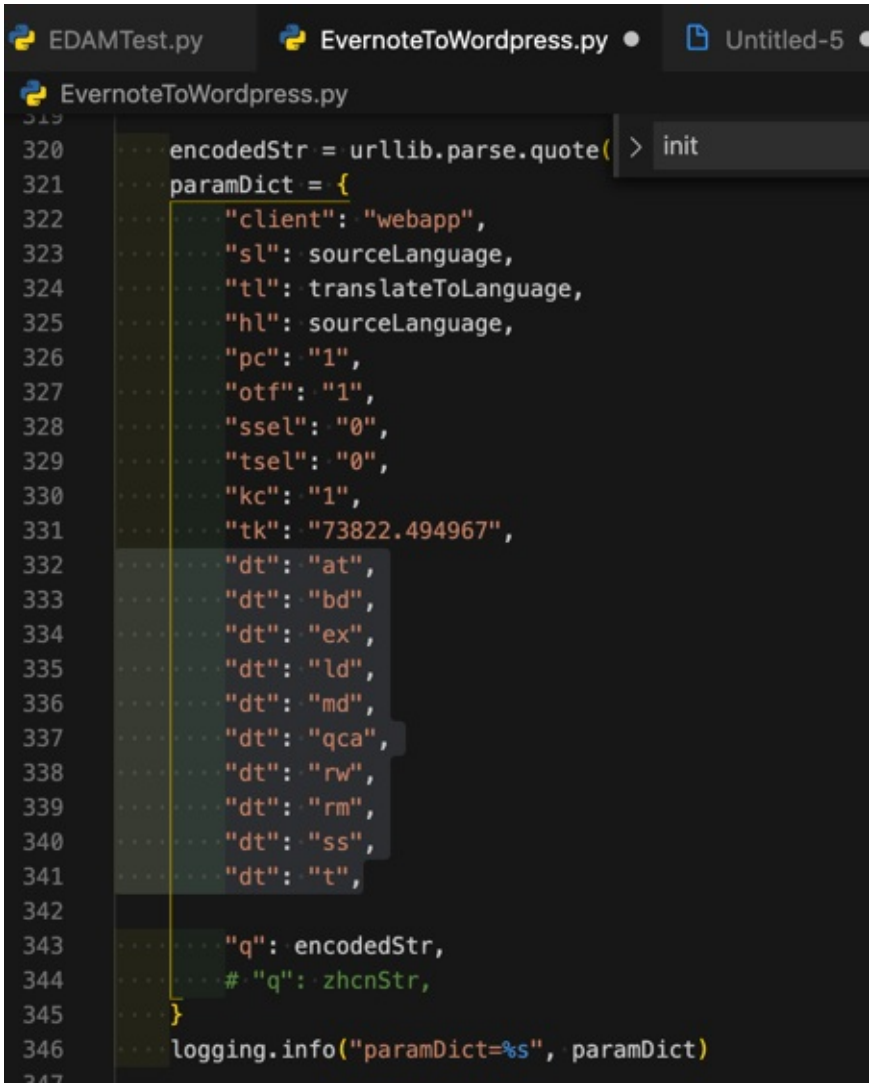


替换成：

```
"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：



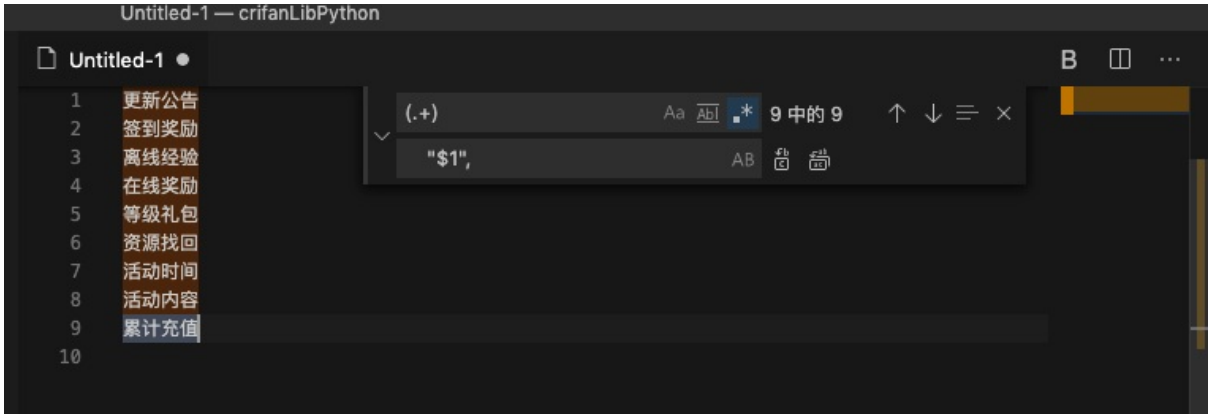
把每个词都加上引号，用于放代码中用

用正则：

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

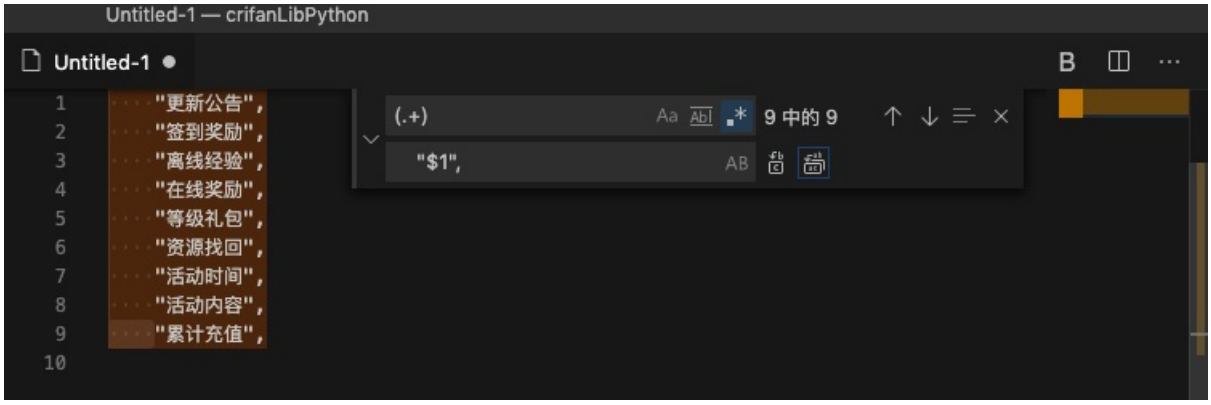
从输入：

```
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
```

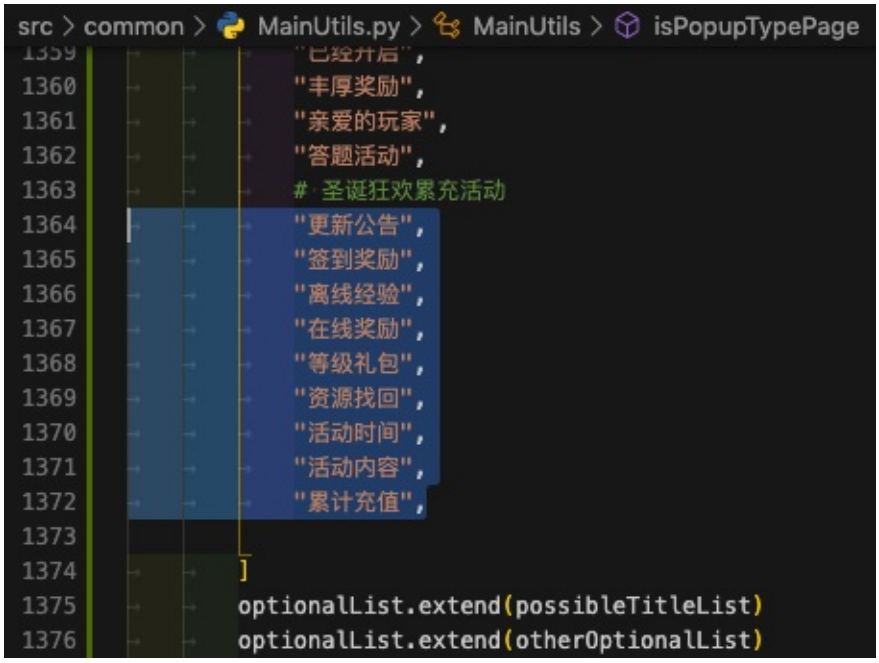


变成：

```
    "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：



去除每行秒及毫秒的多余换行

正则：

```
\n((\d+ s )?\d+ ms)
$1
```

从：

```
completed successfully
3 s 149 ms
Run build
3 s 80 ms
Load build
2 ms
Evaluate settings
1 ms
Finalize build cache configuration


Configure build
420 ms
Load projects
6 ms
Calculate task graph
120 ms
Run tasks
2 s 532 ms
:api:preBuild
21 ms
:api:preDebugBuild
```

```
3 ms
:api:compileDebugAidl
1 s 95 ms
Execute taskAction
869 ms
:cts_provider:preBuild

。。。
```

变成：

```
completed successfully 3 s 149 ms
Run build 3 s 80 ms
Load build 2 ms
Evaluate settings 1 ms
Finalize build cache configuration


Configure build 420 ms
Load projects 6 ms
Calculate task graph 120 ms
Run tasks 2 s 532 ms
:api:preBuild 21 ms
:api:preDebugBuild 3 ms
:api:compileDebugAidl 1 s 95 ms
Execute taskAction 869 ms
:cts_provider:preBuild
。。。
```

即可实现：

去掉了每行多余的换行

Markdown中让EF后缀变下标且加粗

正则：

```
`EF([^`]+)`
EF b sub $1 /sub /b
```

从：

```
    * EF<b><sub>IMPI</sub></b> - IMS private user identity,
    * EF<b><sub>DOMAIN</sub></b> - Home Network Domain Name,
    * `EFIMPU` - IMS Public User Identity (one or more),
    * `EFAD` - Administrative Data (UE operation mode, e.g. normal or type approval),
    * `EFARR` - Access Rule Reference (access rules for files located under the ISIM ADF),
    * `EFIST` - ISIM Service Table (lists available optional services:P-CSCF address, Gene
ric Bootstrapping Architecture (GBA), HTTP Digest, GBA-based Local Key Establishment Mecha
nism, support of P-CSCF discovery for IMS local break out),
```

```
    * `EFP-CSCF` - P-CSCF Address (one or more),
    * `EFGBABP` - GBA Bootstrapping parameters (contains the AKA Random challenge (RAND) a
nd Bootstrapping Transaction Identifier (B-TID) associate with a GBA bootstrapping procedu
re),
    * `EFGBANL` - GBA NAF List (contains the list of NAF_ID and B-TID associated to a GBA
NAF derivation procedure)
    * `EFNAFKCA` - NAF Key Centre Address (one or more).
```

变成：

```
    * EF<b><sub>IMPI</sub></b> - IMS private user identity,
    * EF<b><sub>DOMAIN</sub></b> - Home Network Domain Name,
    * EF<b><sub>IMPU</sub></b> - IMS Public User Identity (one or more),
    * EF<b><sub>AD</sub></b> - Administrative Data (UE operation mode, e.g. normal or type
 approval),
    * EF<b><sub>ARR</sub></b> - Access Rule Reference (access rules for files located unde
r the ISIM ADF),
    * EF<b><sub>IST</sub></b> - ISIM Service Table (lists available optional services:P-CS
CF address, Generic Bootstrapping Architecture (GBA), HTTP Digest, GBA-based Local Key Est
ablishment Mechanism, support of P-CSCF discovery for IMS local break out),
    * EF<b><sub>P-CSCF</sub></b> - P-CSCF Address (one or more),
    * EF<b><sub>GBABP</sub></b> - GBA Bootstrapping parameters (contains the AKA Random ch
allenge (RAND) and Bootstrapping Transaction Identifier (B-TID) associate with a GBA boots
trapping procedure),
    * EF<b><sub>GBANL</sub></b> - GBA NAF List (contains the list of NAF_ID and B-TID asso
ciated to a GBA NAF derivation procedure)
    * EF<b><sub>NAFKCA</sub></b> - NAF Key Centre Address (one or more).
```

Markdown的预览效果：

【此处由于印象笔记冲突导致帖子图片丢失】

去掉其他只保留文件名

正则：

```
\[.+?\]\t(.+.((jpg)|(png)|(gif)|(pdf))).+\n*
$1\n
```

从：

```
[IMG]    主流测试工具分类.jpg    2018-09-21 20:59    159K
[IMG]    内网渗透.png    2020-02-03 01:25    265K
[IMG]    域名搜集途径.png    2018-09-21 20:59    139K
[IMG]    安全漏洞总结.jpg    2018-09-21 20:59    368K
[IMG]    密码找回逻辑漏洞总结.png    2018-12-18 05:09    141K
[IMG]    渗透标准.jpg    2020-02-03 01:25    189K
[IMG]    渗透流程.jpg    2018-09-21 20:59    56K
[IMG]    渗透测试实验室.jpg    2020-02-03 01:25    888K
[IMG]    渗透测试思维导图.png    2020-02-03 01:25    3.7M
```

```
[IMG]    渗透测试流程.jpg    2018-09-21 20:59    221K
[IMG]    渗透测试详细版.jpg    2018-09-21 20:59    260K
[IMG]    社会工程学.jpg    2018-09-21 20:59    95K
[IMG]    系统端口审计琐事.jpg    2018-09-21 20:59    57K
[IMG]    网站入侵图.jpg    2018-09-21 20:59    92K
[IMG]    网络安全绪论.png    2018-09-21 20:59    678K
[IMG]    进阶渗透.png    2018-09-21 20:59    103K
[IMG]    黑客入侵行为分析.gif    2018-09-21 20:59    18K
[IMG]    JavaWeb简介.png    2018-09-21 20:59    913K
[IMG]    Jboss引起的内网渗透.png    2018-09-21 20:59    229K
[IMG]    Maltego使用导图.jpg    2018-09-21 20:59    539K
[IMG]    Nmap.png    2018-09-21 20:59    1.1M
[IMG]    PHP源码审计.png    2018-09-21 20:59    4.0M
[    ]    PTES_MindMap_CN1.pdf    2018-09-21 20:59    417K
[IMG]    Python系统审计.jpg    2018-09-21 20:59    343K
[IMG]    RedTeamManula.jpg    2020-02-03 01:25    5.8M
[IMG]    WEB2HACK.jpg    2020-02-03 01:25    137K
[IMG]    Web安全技术点.jpg    2018-09-21 20:59    193K
[IMG]    Web安全.png    2018-09-21 20:59    228K
[IMG]    Web指纹分析方法.png    2018-09-21 20:59    55K
[IMG]    Web攻击及防御技术.png    2018-09-21 20:59    855K
[IMG]    Web服务器入侵防御.jpg    2018-09-21 20:59    88K
[IMG]    Web 架构中的安全问题.png    2018-09-21 20:59    728K
[IMG]    Windows常见持久控制.png    2020-02-03 01:25    185K
[IMG]    XSS利用架构图.jpg    2020-02-03 01:25    100K
[IMG]    XSS攻击点汇总.png    2018-09-21 20:59    2.2M
[IMG]    nmap.jpg    2018-09-21 20:59    295K
[IMG]    pentest_method.jpg    2018-09-21 20:59    177K
[IMG]    pentester.jpg    2018-09-21 20:59    3.6M
[IMG]    powershell语法.png    2018-09-21 20:59    323K
[IMG]    web应用测试.jpg    2020-02-03 01:25    607K
[IMG]    web渗透.jpg    2018-09-21 20:59    225K
[IMG]    xml安全汇总.png    2018-09-21 20:59    1.7M
```

替换为：

```
主流测试工具分类.jpg
内网渗透.png
域名搜集途径.png
安全漏洞总结.jpg
密码找回逻辑漏洞总结.png
渗透标准.jpg
渗透流程.jpg
渗透测试实验室.jpg
渗透测试思维导图.png
渗透测试流程.jpg
渗透测试详细版.jpg
社会工程学.jpg
系统端口审计琐事.jpg
网站入侵图.jpg
网络安全绪论.png
进阶渗透.png
```

```
黑客入侵行为分析.gif
JavaWeb简介.png
Jboss引起的内网渗透.png
Maltego使用导图.jpg
Nmap.png
PHP源码审计.png
PTES_MindMap_CN1.pdf
Python系统审计.jpg
RedTeamManula.jpg
WEB2HACK.jpg
Web安全技术点.jpg
Web安全.png
Web指纹分析方法.png
Web攻击及防御技术.png
Web服务器入侵防御.jpg
Web 架构中的安全问题.png
Windows常见持久控制.png
XSS利用架构图.jpg
XSS攻击点汇总.png
nmap.jpg
pentest_method.jpg
pentester.jpg
powershell语法.png
web应用测试.jpg
web渗透.jpg
xml安全汇总.png
```

搜索get函数被调用的地方

默认如果搜

```
get(
```

则会搜出来很多，有很多个 我们不希望看到的

```
http.get(
```

想要排斥掉 `http.get(`

想到了用正则

参考之前语法：

`(?<!xxx): negative look behind (assertion)=反向否定断言`

去用：

```
(?<!http\.)get\(
```

然后只有7个，都是我们要找到了：

【此处由于印象笔记冲突导致帖子图片丢失】

想要搜索15秒相关的内容

直接搜15，找到很多 `xxx="15"` 不是我们要的：

【此处由于印象笔记冲突导致帖子图片丢失】

所以想要排除掉：

- 前面是双引号
- 后面也是双引号的

所以用：

```
(?<!")15(?!")
```

即可过滤掉，找到其他地方的15：

【此处由于印象笔记冲突导致帖子图片丢失】

即，支持用

- `look ahead negative`
- `look behind negative`

去过滤掉不要的情况。

把链接标题和地址变成Markdown中url格式

用：

```
^([^/]+)\n(https?://.+)
* [$1]($2)
```

把：

```
PyCharm Community Edition and Professional Edition Explained: Licenses and More | PyCharm Blog
https://blog.jetbrains.com/pycharm/2017/09/pycharm-community-edition-and-professional-edition-explained-licenses-and-more/
Differences between Pycharm community and professional edition?: Python
https://www.reddit.com/r/Python/comments/7370yp/differences_between_pycharm_community_and/
Pycharm的教育版和社区版有什么区别？ - 知乎
https://www.zhihu.com/question/47511825
使用PyCharm进行Python远程调试
http://leoc.leanote.com/post/remote-debugging-with-pycharm
Pycharm的远程代码编辑 - kiwik's blog
http://kiwik.github.io/python/2013/08/12/Pycharm的远程代码编辑/
```

变成markdown中url的格式：

```
* [PyCharm Community Edition and Professional Edition Explained: Licenses and More | PyCharm Blog](https://blog.jetbrains.com/pycharm/2017/09/pycharm-community-edition-and-professional-edition-explained-licenses-and-more/)
* [Differences between Pycharm community and professional edition?: Python](https://www.reddit.com/r/Python/comments/7370yp/differences_between_pycharm_community_and/)
* [Pycharm的教育版和社区版有什么区别？ - 知乎](https://www.zhihu.com/question/47511825)
* [使用PyCharm进行Python远程调试](http://leoc.leanote.com/post/remote-debugging-with-pycharm)

* [Pycharm的远程代码编辑 - kiwik's blog](http://kiwik.github.io/python/2013/08/12/Pycharm的远程代码编辑/)
```

单行变多行加换行

想要从 服务中心_查询保险条款题-条款内容页 复制出的文字：

```
【碰撞】指被保险机动车或其符合装载规定的货物与外界固态物体之间发生的、产生撞击痕迹的意外撞击。 【倾覆】指被保险机动车由于自然灾害或意外事故，造成本被保险机动车翻倒，车体触地，失去正常状态和行驶能力，不经施救不能恢复行驶。 【坠落】 指被保险机动车在行驶中发生意外事故，整车腾空后下落，造成本车损失的情况。非整车腾空，仅由于颠簸造成被保险机动车损失的，不属于坠落。 【外界物体倒塌】指被保险机动车自身以外的物体倒下或陷下。 【自燃】指在没有外界火源的情况下，由于本车电器、线路、供油系统、供气系统等被保险机动车自身原因或所载货物自身原因起火燃烧。 【火灾】指被保险机动车本身以外的火源引起的、在时间或空间上失去控制的燃烧（即有热、有光、有火焰的剧烈的氧化反应）所造成的灾害。 【次生灾害】指地震造成工程结构、设施和自然环境破坏而引发的火灾、爆炸、瘟疫、有毒有害物质污染、海啸、水灾、泥石流、滑坡等灾害。 【暴风】指风速在28.5米/秒（相当于11级大风）以上的大风。风速以气象部门公布的数据为准。 【暴雨】指每小时降雨量达16毫米以上，或连续12小时降雨量达30毫米以上，或连续24小时降雨量达50毫米以上。 【洪水】指山洪暴发、江河泛滥、潮水上岸及倒灌。但规律性的涨潮、自动灭火设施漏水以及在常年水位以下或地下渗水、水管爆裂不属于洪水责任。 【玻璃单独破碎】指未发生被保险机动车其他部位的损坏，仅发生被保险机动车前后风挡玻璃和左右车窗玻璃的损坏。 【车轮单独损坏】指未发生被保险机动车其他部位的损坏，仅发生轮胎、轮辋、轮毂罩的分别单独损坏，或上述三者之中任意二者的共同损坏，或三者的共同损坏。 【车身划痕损失】仅发生被保险机动车车身表面油漆的损坏，且无明显碰撞痕迹。 【新增设备】指被保险机动车出厂时原有设备以外的，另外加装的设备和设施。 【新车购置价】指本保险合同签订地购置与被保险机动车同类型新车的价格，无同类型新车市场销售价格的，由投保人与保险人协商确定。 【单方肇事事故】指不涉及与第三者有关的损害赔偿的事故，但不包括自然灾害引起的事故。 【家庭成员】指配偶、子女、父母。 【市场公允价值】 指熟悉市场情况的买卖双方在公平交易的条件下和自愿的情况下所确定的价格，或无关联的双方在公平交易的条件下一项资产可以被买卖或者一项负债可以被清偿的成交价格。
```

变成易读的，多行。

正则：

```
(【[^【】]+】)
\n$1
```

把：

【此处由于印象笔记冲突导致帖子图片丢失】

变成：

```
【碰撞】指被保险机动车或其符合装载规定的货物与外界固态物体之间发生的、产生撞击痕迹的意外撞击。
【倾覆】指被保险机动车由于自然灾害或意外事故，造成本被保险机动车翻倒，车体触地，失去正常状态和行驶能力，不经施救不能恢复行驶。
【坠落】 指被保险机动车在行驶中发生意外事故，整车腾空后下落，造成本车损失的情况。非整车腾空，仅由于颠簸造成被保险机动车损失的，不属于坠落。
【外界物体倒塌】指被保险机动车自身以外的物体倒下或陷下。
【自燃】指在没有外界火源的情况下，由于本车电器、线路、供油系统、供气系统等被保险机动车自身原因或所载货物自身原因起火燃烧。
【火灾】指被保险机动车本身以外的火源引起的、在时间或空间上失去控制的燃烧（即有热、有光、有火焰的剧烈的氧化反应）所造成的灾害。
【次生灾害】指地震造成工程结构、设施和自然环境破坏而引发的火灾、爆炸、瘟疫、有毒有害物质污染、海啸、水灾、泥石流、滑坡等灾害。
【暴风】指风速在28.5米/秒（相当于11级大风）以上的大风。风速以气象部门公布的数据为准。
【暴雨】指每小时降雨量达16毫米以上，或连续12小时降雨量达30毫米以上，或连续24小时降雨量达50毫米以上。
【洪水】指山洪暴发、江河泛滥、潮水上岸及倒灌。但规律性的涨潮、自动灭火设施漏水以及在常年水位以下或地下渗水、水管爆裂不属于洪水责任。
【玻璃单独破碎】指未发生被保险机动车其他部位的损坏，仅发生被保险机动车前后风挡玻璃和左右车窗玻璃的损坏。
【车轮单独损坏】指未发生被保险机动车其他部位的损坏，仅发生轮胎、轮辋、轮毂罩的分别单独损坏，或上述三者之中任意二者的共同损坏，或三者的共同损坏。
【车身划痕损失】仅发生被保险机动车车身表面油漆的损坏，且无明显碰撞痕迹。
【新增设备】指被保险机动车出厂时原有设备以外的，另外加装的设备和设施。
【新车购置价】指本保险合同签订地购置与被保险机动车同类型新车的价格，无同类型新车市场销售价格的，由投保人与保险人协商确定。
【单方肇事事故】指不涉及与第三者有关的损害赔偿的事故，但不包括自然灾害引起的事故。
【家庭成员】指配偶、子女、父母。
【市场公允价值】 指熟悉市场情况的买卖双方在公平交易的条件下和自愿的情况下所确定的价格，或无关联的双方在公平交易的条件下一项资产可以被买卖或者一项负债可以被清偿的成交价格。
```

方便拷贝到别处使用和阅读了。

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