**2020学年云计算大作业实验报告**

**组号：17 小说网站数据**

**小组成员信息：**

组长：冯泊涓 软件学院 181250030

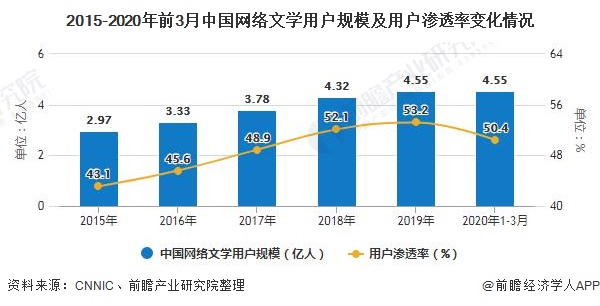
组员：吴昊宇 软件学院 181250154

王馨逸 软件学院 181250144

张嘉玥 软件学院 181250185

**应用场景**

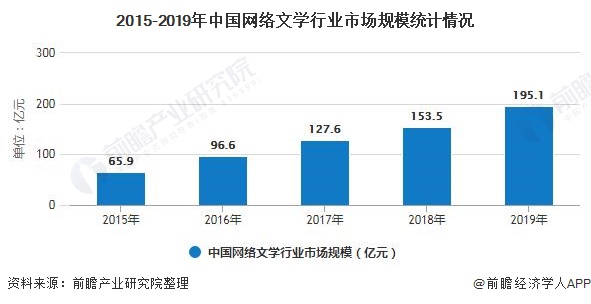
网络写作的意义超过了以往任何一次文学革命。中国当代文学发展30年最重要的变化，就是网络文学的出现。网络写作经过15年的发展，已经不单单是一种文化现象，它已成为了一种社会形态。



截至2019年末，中国网络文学用户规模达4.55亿人，同比 2018年增长 5.32%。从数据表现来看，网络文学用户规模依然呈现出**稳定增长**的健康发展态势。

2020年第三季度，中国网络文学用户渗透率(网络文学用户数量/网民数量)为50.4%，较2019年末有小幅下滑，主要是因为**网民规模增长速度高于网络文学用户增长速度**。

近年来，我国数字内容**版权环境持续优化**，网络文学用户**付费意识**也逐渐增强，推动网络文学行业创造更多有价值的作品，进而推动国内网络文学业务蓬勃发展。



网络文学行业的盈利模式主要来源于付费阅读、版权售卖和广告收入等形式，近年来**网络视频**的火热，让**网络文学的版权营收**持续提升，整个行业市场规模增长非常迅速，2019年我国网络文学市场规模约为195.1亿元。

**待解决的业务需求**

我们选取了晋江文学城作为研究对象，在以下方面进行了数据探究：

* 小说标签分类统计（发表时间-小说标签-热度）
* 小说主角情感与性取向 （发表时间-小说类别）
* 小说人物常用姓、名统计 （发表时间-主角姓名-配角姓名）（小说时代-主角姓名-配角姓名）

由此，我们可以：

* 了解10年来网络作品类别的热门趋势变化，从而间接地了解10年来网络文学阅读群体的特征变化
* 管中窥豹，由网络文学作品的统计数据映射一部分的社会现象与文化偏好
* 在创作自己的网络文学作品时，获得选题内容上的潮流趋势参考
* 在创作自己的网络文学作品时，解决起名废问题（x）

**待回答的研究问题**

* 网民们喜欢阅读什么类型的网络作品呢？
* 十年来，网民的阅读偏好有没有发生什么变化呢？
* 十年来，有没有什么隐含在网络文学里的社会现象？
* 作者们喜欢给笔下的人物取什么样的名字?

**数据来源与数据内容**

数据来源：<http://my.jjwxc.net/>

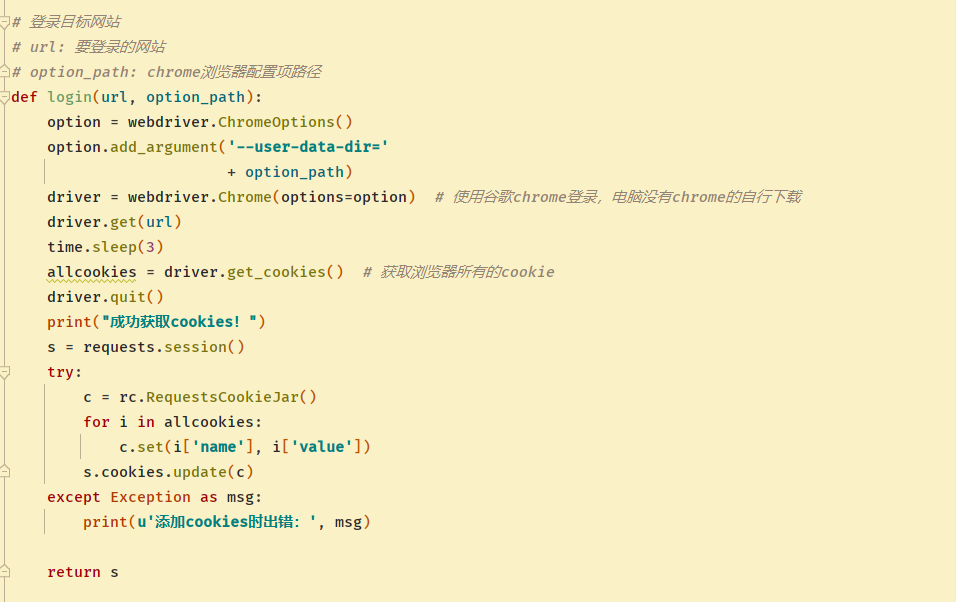
**数据内容**

2010-2020每一年晋江文学城作品按收藏顺序排序的前200页的每一部作品的作品信息，包含：

* author: 作者
* name: 书名
* web\_url: 书籍主页
* originality: 原创性
* disposition: 性向
* times: 时代
* type: 类型
* style: 风格
* word\_number: 字数
* score: 作品积分
* pubdate: 发表时间
* tags: 标签
* leading: 主角名
* supporting: 配角名

**数据获取方式**

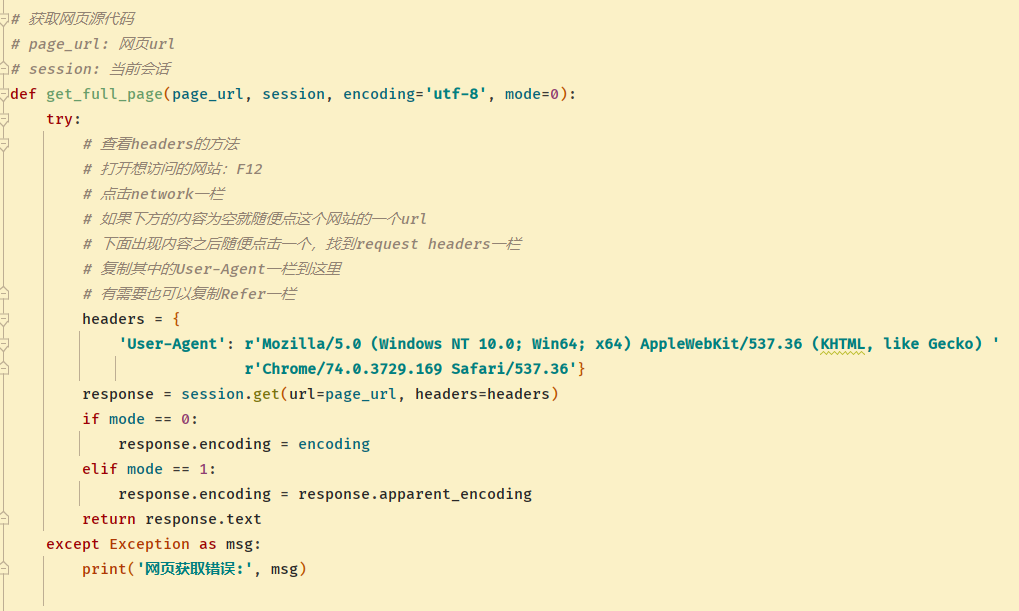
**创建会话**



由于晋江文学城网站访问有限制，需要登录才能无限制访问所有子网页。因此本组使用的方法是先手动登录晋江文学城并保持登录状态，再使用selenium包中的webdriver模块控制浏览器访问晋江文学城。这样，webdriver模块登陆的晋江文学城就是已经登录的状态。接着获取浏览器产生的cookies，并使用该cookies创建会话（requests.session()）并在数据爬取的过程中一直保持会话，就可以无限制爬取数据。

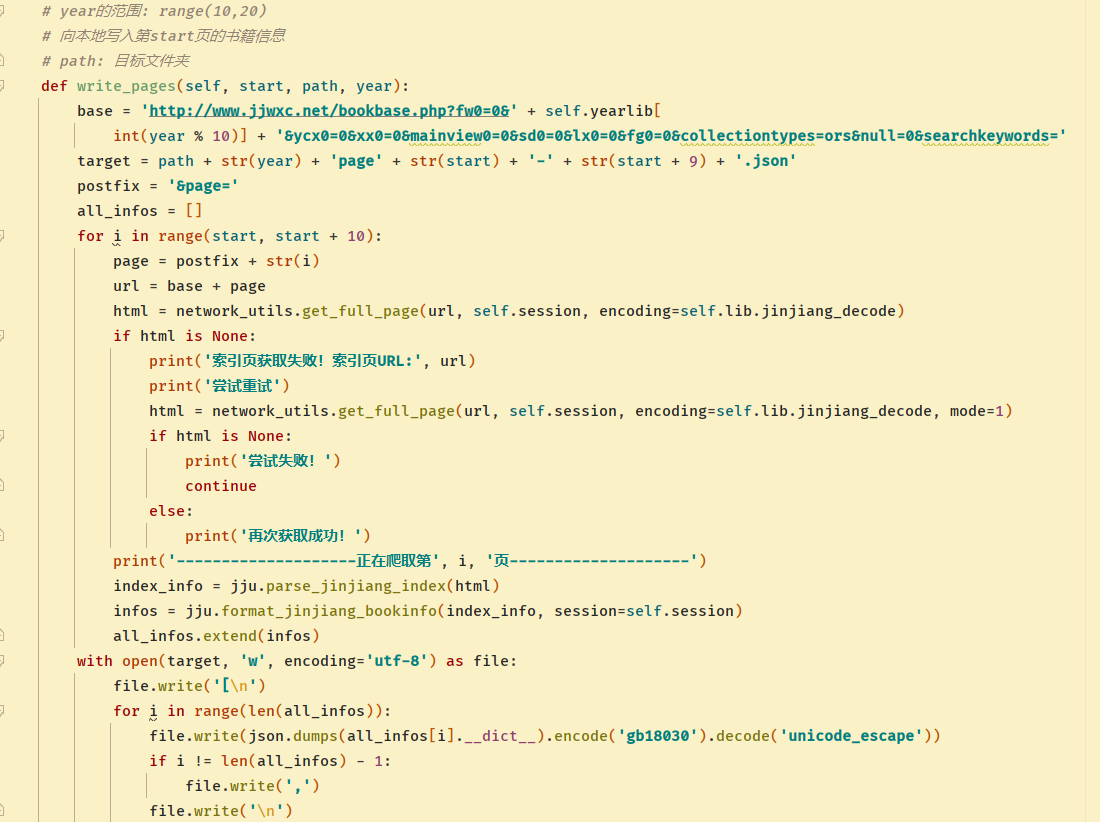
图中的代码是获取cookies创建会话并返回会话的方法。

**获取网页源码**



使用上个步骤创建的会话获取网页源码（session.get()），并对获取到的源码进行编码再返回。

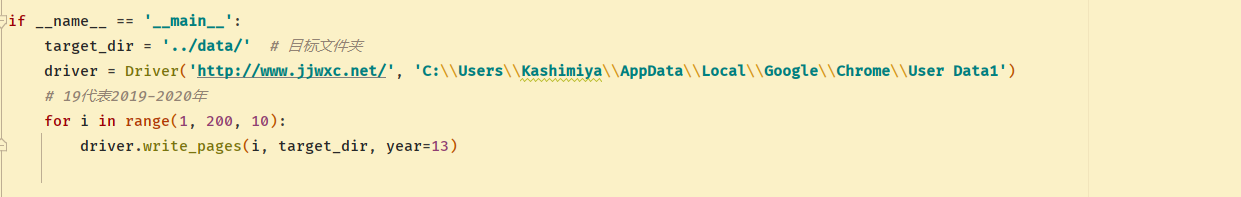
**写文件**





这个方法是整个爬取数据程序的驱动方法。这个代码爬取从start开始10页索引页的数据，并以json的格式保存到target。需要注意的是，由于是中文数据，所以保存文件时用了gb18030编码。

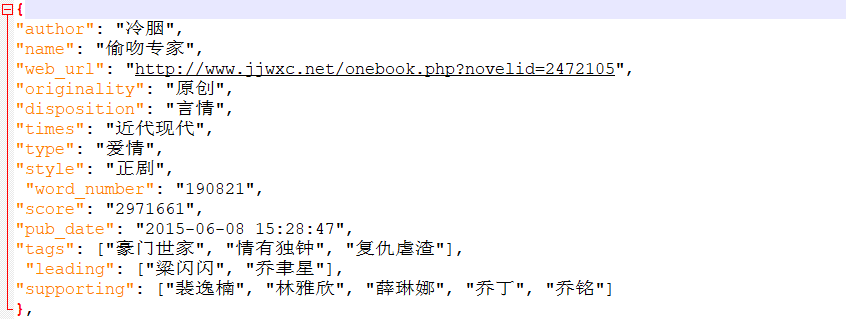
**运行**



参数：

* target\_dir: 文件输出目标文件夹
* Driver(param1, param2):
* param1: 需要建立会话的网址
* param2: chrome用户设置的位置（为了webdriver打开网页时，能够保持晋江的登陆状态）
* range(x, y, 10): 爬取索引页x到y页的数据，每10页保存一次（如果一定需要更改10页保存一次的设置，则要更改代码中其他部分）
* year: 爬取的年份。范围：10-19（19代表的是2019和2020两年）

**数据格式**



**存储方式**：json文件本地存储

**预处理过程**

**网页内容分析、提取、预处理**

获取晋江文学城一本书的信息主要用到了两个页面：

1. 书籍索引页



可以看出是表格形式。整体表格结构的代码如下



其中，一本书的表格项的源码示例如下



可以看到，整体表格结构的代码是一个table标签，class是cytable；而一本书中需要的标签有两类：a和td。针对这个规律，爬取代码如下



使用BeautifulSoup包对源码进行解析，根据规律筛选出需要的a和td标签分别存放在names\_a和names\_td两个数组中一起传回。

这里得到的数据种类有作者、书名、书籍首页、原创性、时代、类型、风格、是否完结、字数、作品积分、发表时间。

1. 书籍详情页

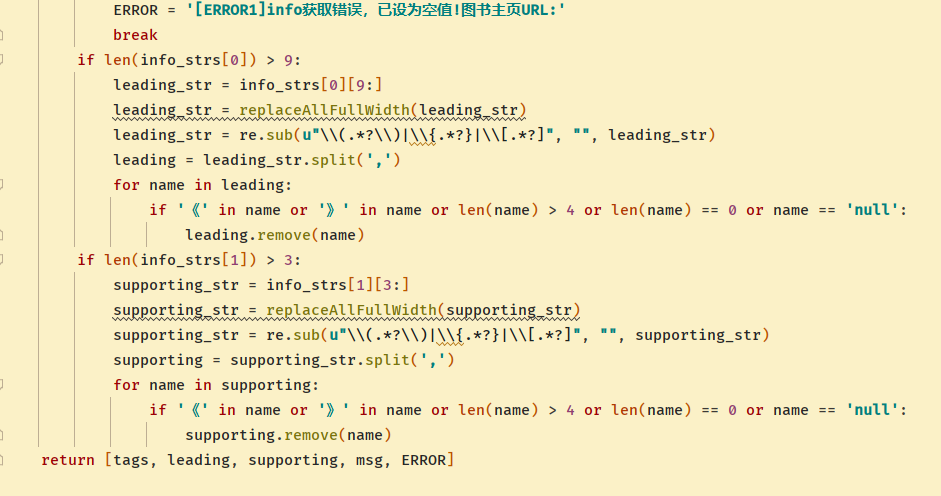


上面已经提到，书籍索引页可以获取到书籍首页的url。所以，在爬取、分析完索引页源码后，使用会话就可以获取书籍详情页代码从而获取到更多信息。图中可以看到下方三排有我们需要的信息（标签、主角、配角）。这一段文字的源码如下

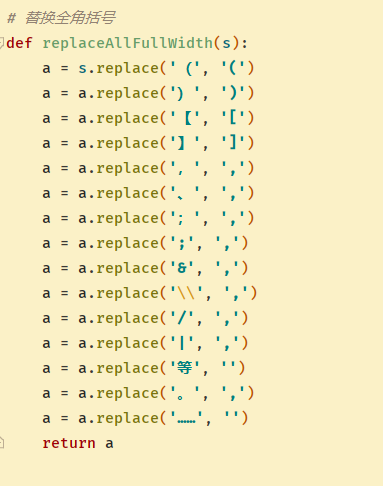


可以看到，我们需要的信息包裹在一个class为smallreadbody的div标签下。其中，内容标签的每一项都在一个style为text-decoration:none;color: red;的a标签下；主角和配角都在一个class为bluetext的span标签下。根据这些规律，爬取代码如下所示。

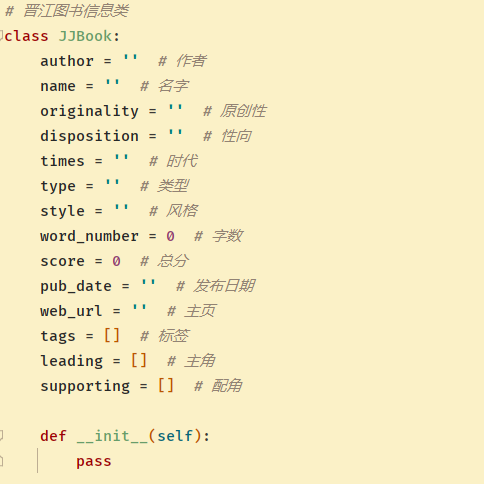




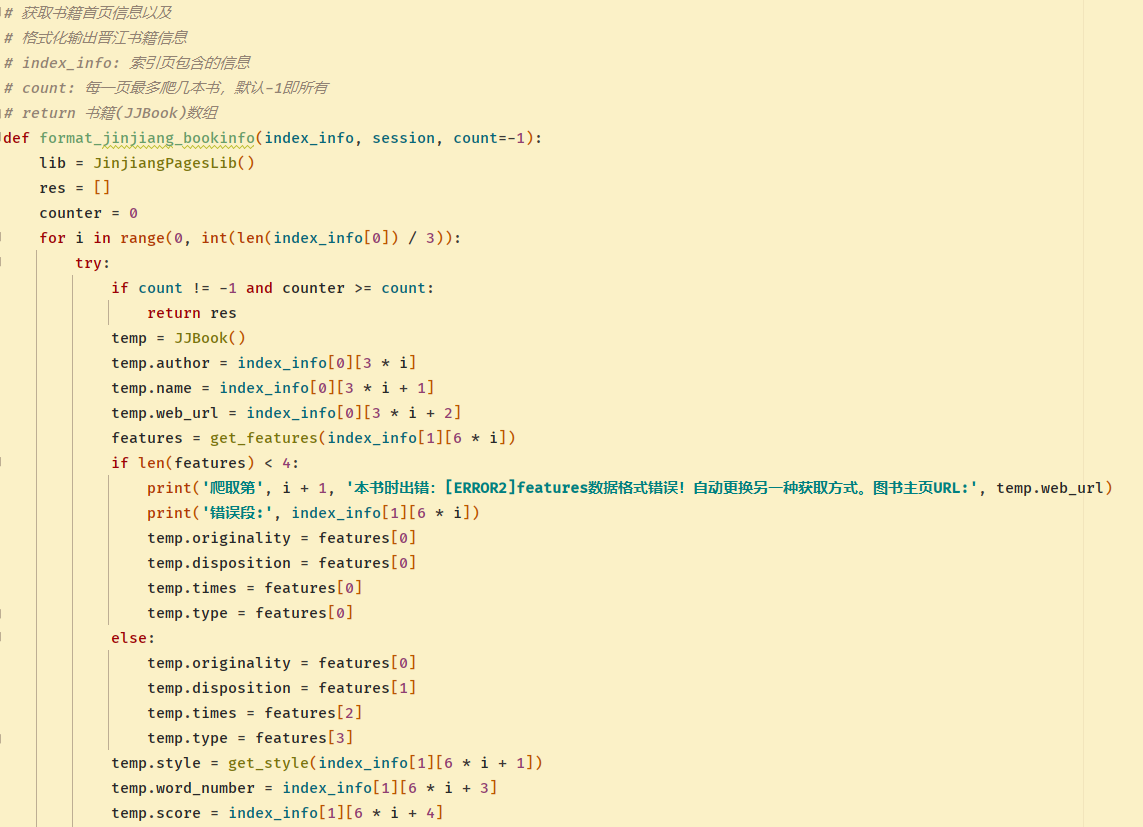
将获取到的信息分别装在tags, leading, supporting数组中，加上返回信息（msg，ERROR，主要是为了报错和debug）一并返回。此外，这段代码还将所有的中文符号转化为了英文逗号（replaceAllFullWidth()方法，如下所示。实际上为了便于split，还将很多奇怪的符号全部替换成了英文逗号，还删除了“等”“……”这样的不属于名字的成分），并删除掉了括号中的内容，再将主角配角名做了分割。

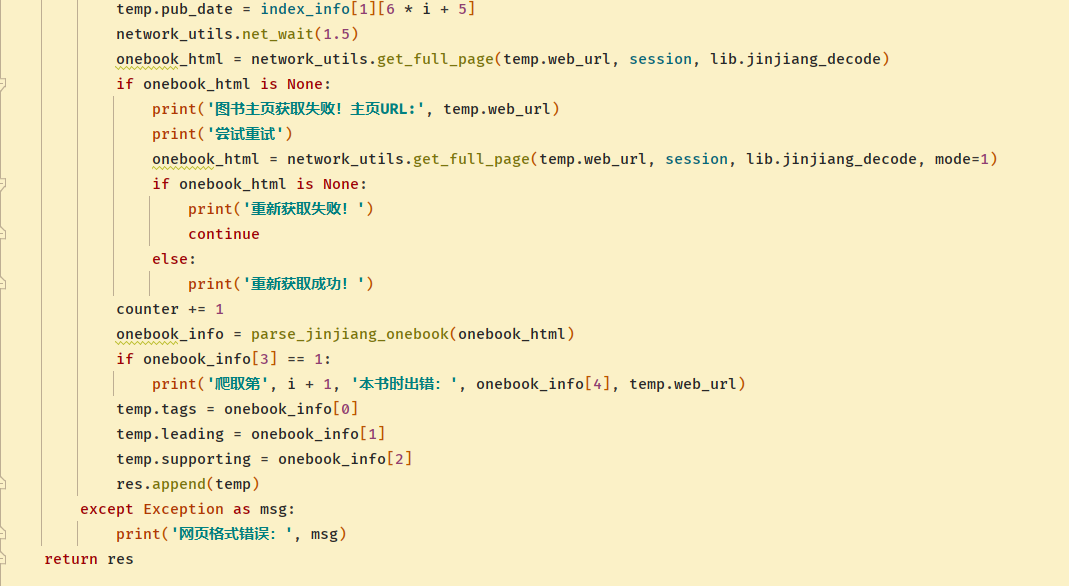


1. 接着便是对上面两种页面得到的数据进行整合、结构化。首先，结构化python类如下所示



接着，如下所示，是整合数据的代码

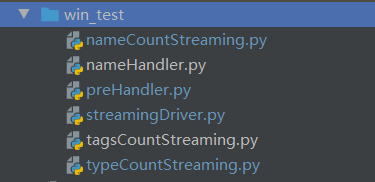




这个方法调用了之前所说的两个方法（parse\_jinjiang\_index()、parse\_jinjiang\_onebook()）来分别获取索引页数据（index\_info）和书籍首页数据（onebook\_info）。随后将获取到的每一本书数据分别装入一个书籍信息对象（JJBook）中，最后将这个书籍信息对象的数组（res）返回。

**流计算总体架构**

* 流计算部分使用了pyspark语言，代码共有6个文件，包括：
* 数据预处理和流构造：preHandler.py
* 流计算：nameCountStreaming.py，tagsCountStreaming.py，typeCountStreaming.py
* 姓名处理工具：nameHandler.py
* 运行驱动器：streamingDriver.py



**流准备和监听**

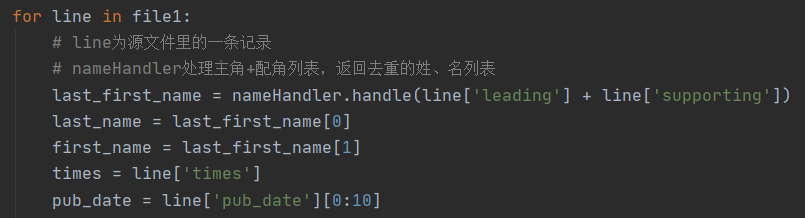
在爬虫已经获取到源文件之后，我们希望通过文件的转存来构造导入监听文件夹的数据流。同时，由于源文件的数据结构复杂程度远超探究命题所需，我们还需要对爬取的源文件按照命题的需求进行筛选和整理。对此，我们使用一个preHandler预处理程序来完成数据整理和流构造的任务。

**preHandler.py**

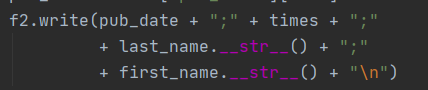
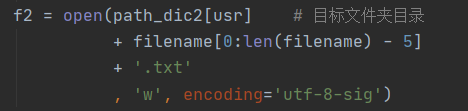
在preHandler中，一共包含了4个方法：getFileList、nameCount、typeCount、tagCount。其中，getFileList是公用工具，负责读取所有的源文件名，方便源文件的遍历处理。nameCount、typeCount、tagCount分别对三个探究命题进行针对性的数据处理和文件导出。

**①nameCount方法：**

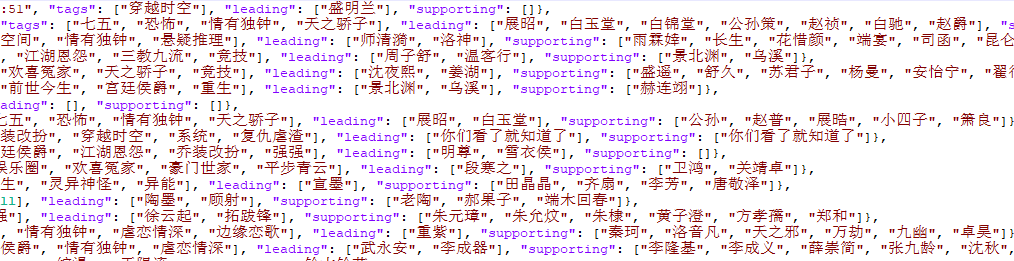
nameCount方法筛选了源文件中：pub\_date发表日期，times故事背景时代，leading主角，supporting配角，共4个字段。其中，pub\_date字段的发表时间精确度达到了时分秒，超过我们的需求；我们只截取保留了它的年月日部分。



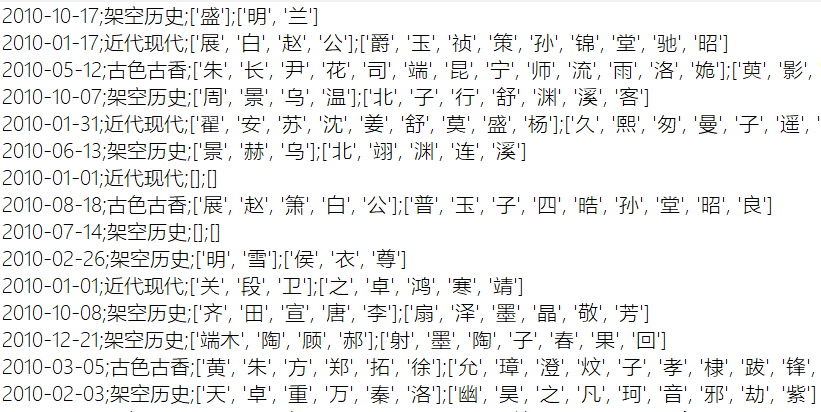
由于后续streaming处理中，需要按行读取处理文本数据，这里放弃了使用json进行结构化的存储，而是考虑使用csv文件或txt文件进行文本存储。由于姓、名涉及列表结构，包含“，”分隔符，与csv文件的默认分隔符重复，我们最终选用了txt格式进行存储。



最终处理结果如图所示：



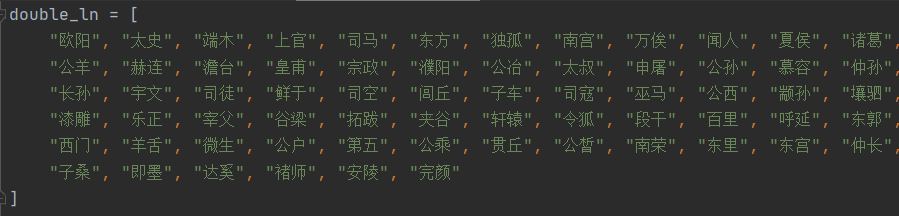
处理前的角色名列表



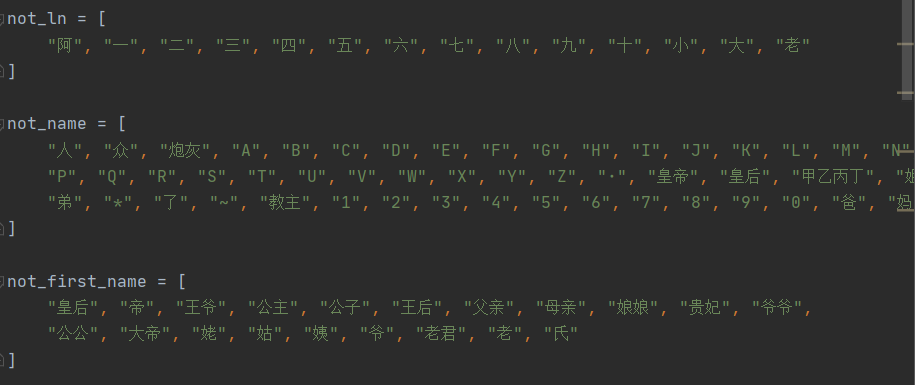
处理后的角色名列表

**②nameHandler.py**

在nameCount方法中，leading和supporting列表调用了nameHandler进行了处理，将角色名转换为姓、名列表。nameHandler中包括复姓的划分、姓名去重、非法姓名判断等。

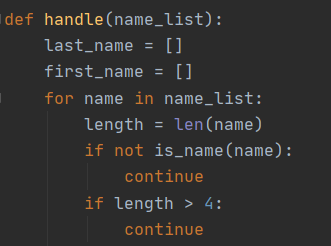


常见复姓列表

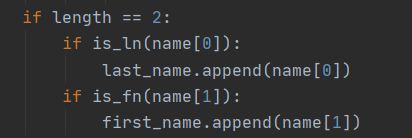
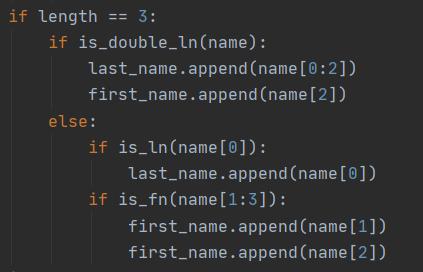
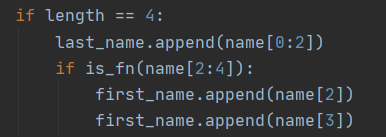


非法姓名列表

在获得传入的角色名列表后，我们首先从长度、内容上进行了非法姓名判断。如果姓名内容包含not\_name列表中列举的非法标识符、或姓名长度超过4个字，都将被判断为非法内容，直接过滤。



接下来我们对4、3、2字角色名作了姓、名区分。对4字名，我们直接划分为复姓+名的组合；对3字名，我们首先判断是否为复姓+单名，如果不是，则划分为单姓+名的组合；对2字名，我们直接划分为姓+名的组合。对于每一个名字，我们都做了姓、名的非法检查。



遍历获得姓、名列表后，我们意识到同一个作者可能会在主配角列表中大量复用同一个姓氏、辈分字，很可能影响字的权重。为此，我们对列表进行了去重操作，希望每一个作者的取名字典都是平等的。

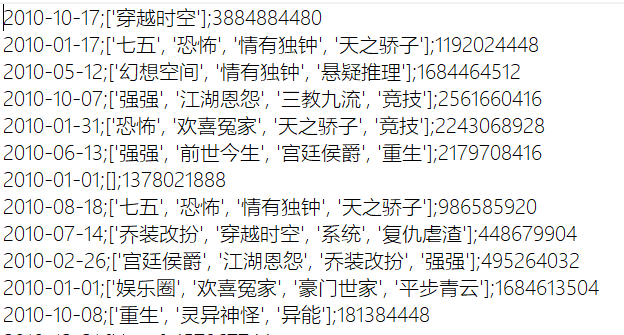


**③typeCount、tagCount方法：**

typeCount、tagCount和nameCount总体的逻辑是一致的，只是选取的源数据字段不同。其中，typeCount选取了作品主角情感与性向分类、发表时间；tagCount选取了作品标签、发表时间、热度。







**流计算**

在流计算部分，我们给三个命题分别安排了不同的流处理文件：

nameCountStreaming.py：按照发表时间、作品时代分别统计姓、名使用频率

typeCountStreaming.py：按照发表时间、作品性向统计作品数

tagsCountStreaming.py：按照发表时间、作品标签统计标签热度

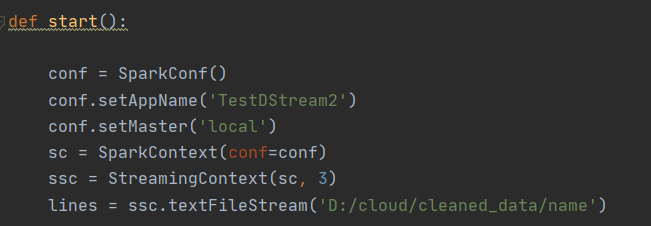
三个文件分别监听不同的文件夹，为preHandler中对应方法的目标文件夹。

最后，我们使用了一个streamingDriver驱动器，以双线程同时启动preHandler和streaming，达到数据流的投放和监听效果。

**①nameCountStreaming.py**

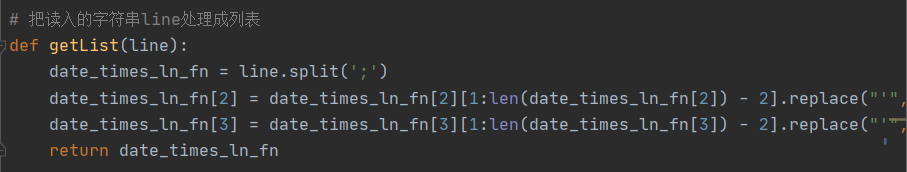
本文件的统计结果有2份，分别为：根据发表年份统计的热门名、根据作品时代统计的热门姓。两者的处理方式相同，这里仅仅解说根据发表日期统计热门名的过程。

首先，我们设置了spark的相关信息。由于本份代码是在win10操作系统上实验的测试代码，所以监听文件夹是win10操作系统下的文件夹。

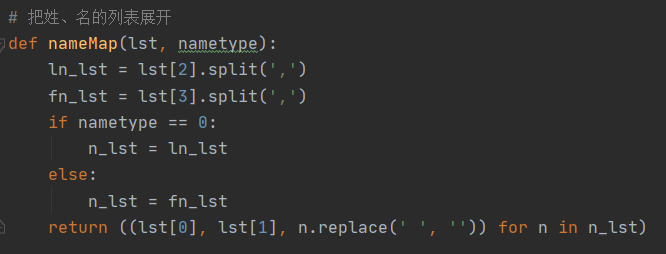
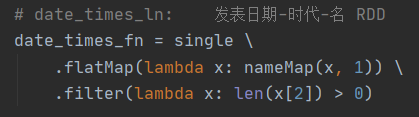


监听到的新文件被读取到lines中，我们调用了getList方法对lines进行处理，将文本处理为列表，格式为【发表日期，作品时代，姓列表，名列表】

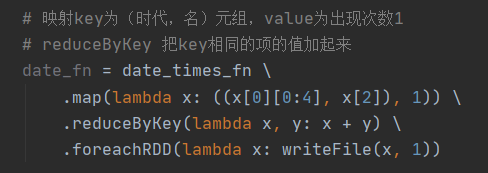




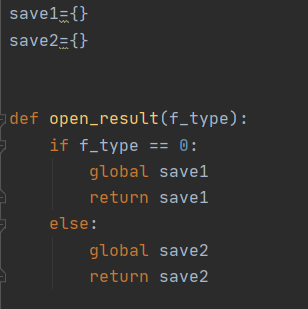
随后，为了统计热门名，我们需要将名列表展开。这里首先使用了flatMap操作，将一个RDD映射为多个RDD，映射方法由nameMap定义；然后使用fliter进行过滤，把名为空的RDD删去。nameMap映射方法为姓、名映射公用，nametype参数负责姓、名的选择。



获得格式为【发表时间，作品时代，名】的RDD后，我们希望统计同一作品时代下名的出现次数。这里我们使用map方法，将RDD映射为【（作品时代，名），1】的格式；然后使用reduceByKey方法进行计数统计。

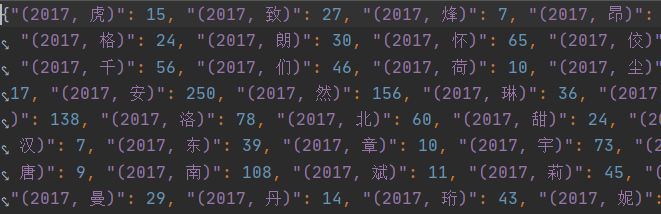


值得一提的是统计结果的写入：流处理只能按照时间分片逐片统计生成统计结果，需要自行解决结果整合的问题。在这里，我们使用了foreachRDD方法，调用结果文件的写入方法writeFile。writeFile负责读取全局变量中存储的历史统计结果，将本次处理结果与历史统计结果进行整合，再写回。



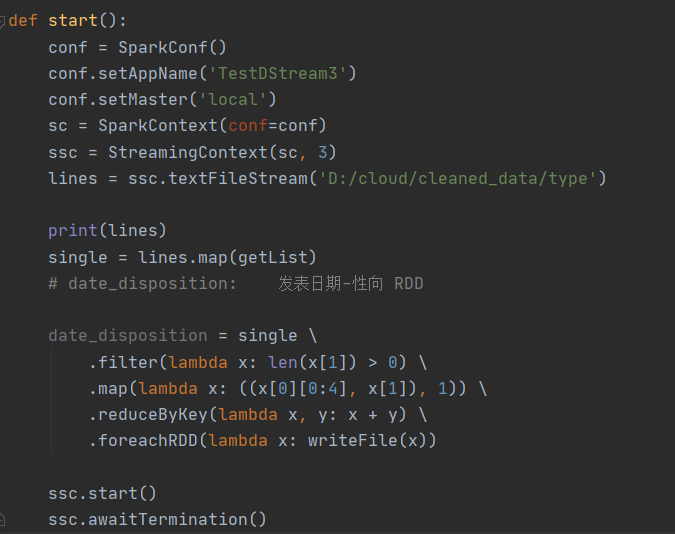


流处理最终得到的结果文件如下：

­

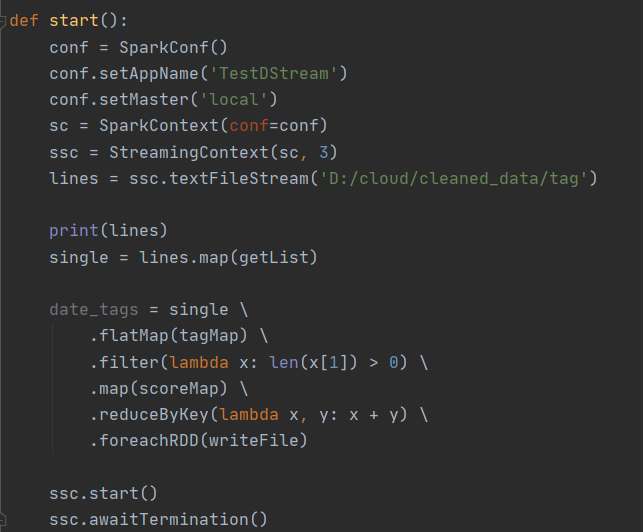
**②typeCountStreaming.py**

整体的处理过程基本一致，依次使用了map(getList)将RDD由文本内容转为结构化内容；filter过滤空项；map匹配key和value以便统计计数；reduceByKey进行同key统计；最后使用foreachRDD(writeFile)进行结果整合写入。



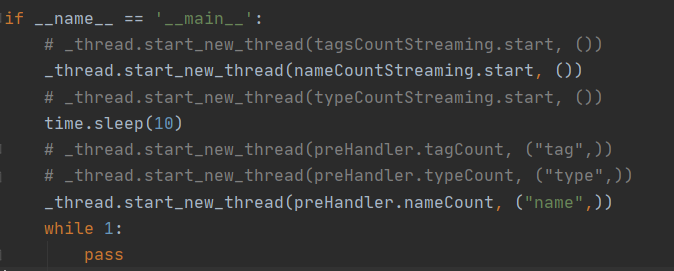
**③tagsCountStreaming.py**

整体的处理过程基本一致，依次使用了map(getList)将RDD由文本内容转为结构化内容；flatMap将标签列表映射展开为单项；filter过滤空项；map匹配key和value以便统计计数；reduceByKey进行同key统计；最后使用foreachRDD(writeFile)进行结果整合写入。



**④streamingDriver.py**

为了达到数据流实时投放、实时监听的效果，我们需要将preHandler和streaming先后启动。我们利用python多线程将这一操作进行了自动化。



在运行过程中，我们发现python多线程与java多线程的不同：当python的主线程的任务结束后，主线程会“自戮”，子线程也会因此而结束。为了保证线程持续运行，我们给main函数添加了while 1死循环以维持它的生命周期，等待流处理结束后再手动关闭进程。

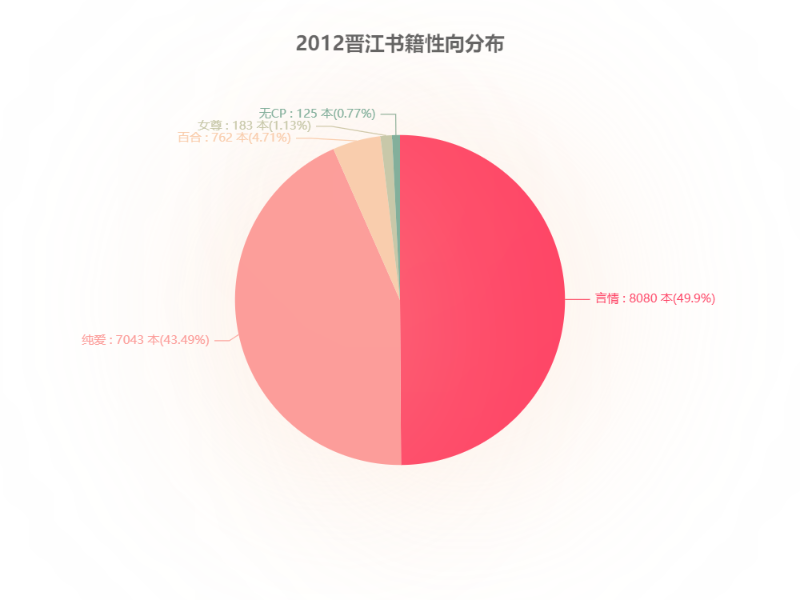
**结果动态展示**

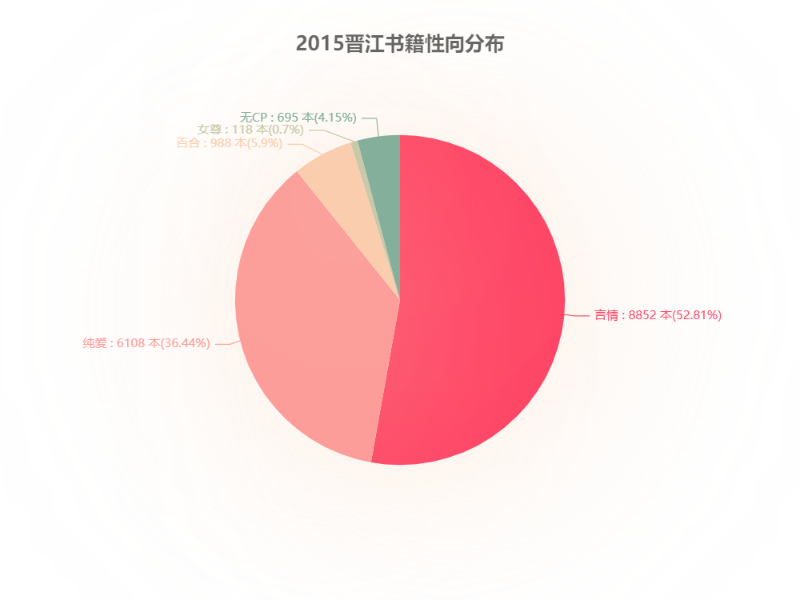
**饼图、折线图——不同年份的晋江书籍性向分布**

* 使用echarts模块
* 文件
  + ./EchartsPre/pre.html 2010-2020年各性向占比饼图
  + ./EchartsPre/prePlus.html 各性向2010-2019年数量变化折线图
* 浏览方法
  + 双击html文件即可使用浏览器浏览
  + 使用右上角按钮即可切换到下一年/下一性向



* 预览（图片使用echarts内置的下载功能下载，因此看不到右上角按钮）
  + 饼图





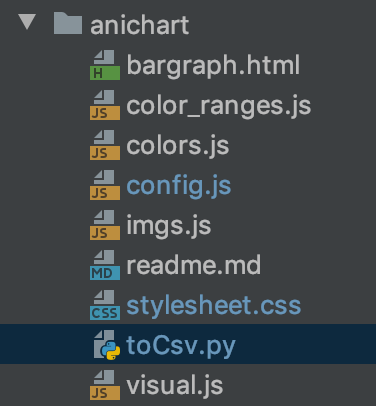
* + 折线图



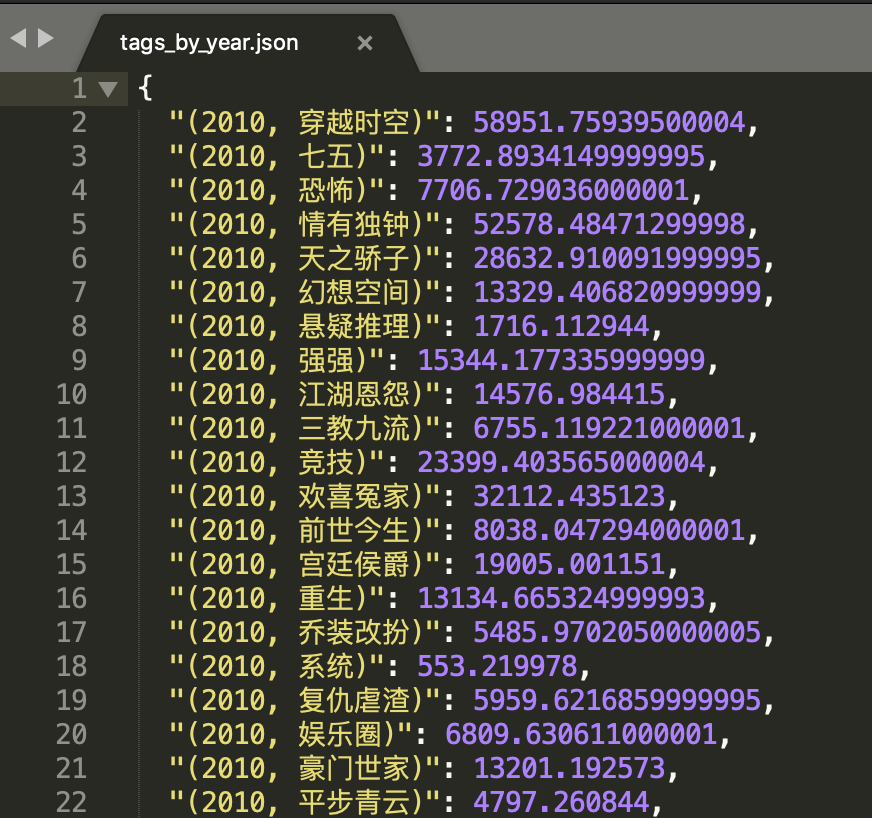


**动态柱状图——书籍标签热度的年份变化趋势**

Anichart模块文件结构如图：



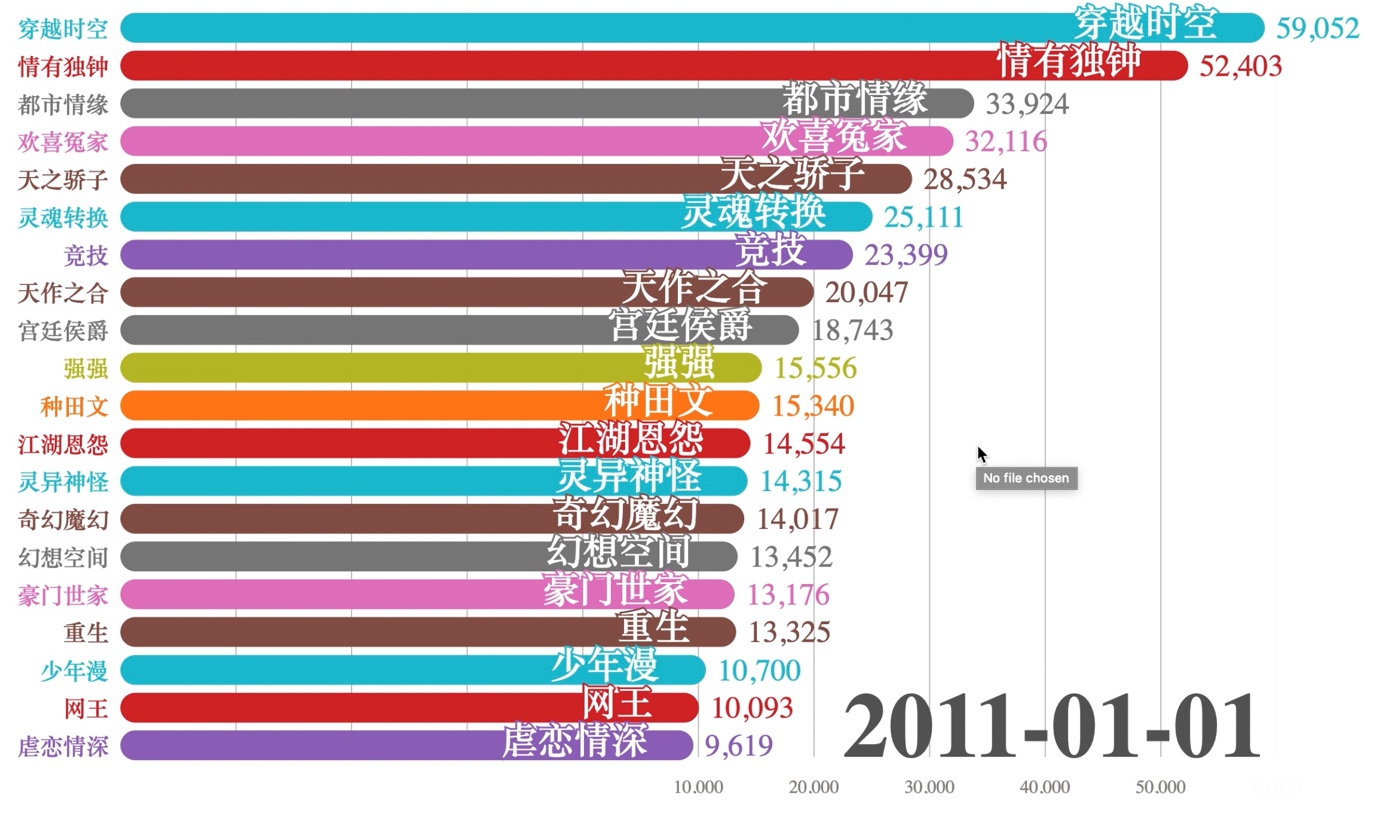
在流处理的过程中，输出了结果文件 tags\_by\_year.json文件，文件格式为”(年份，tag)”: 热度，如下图

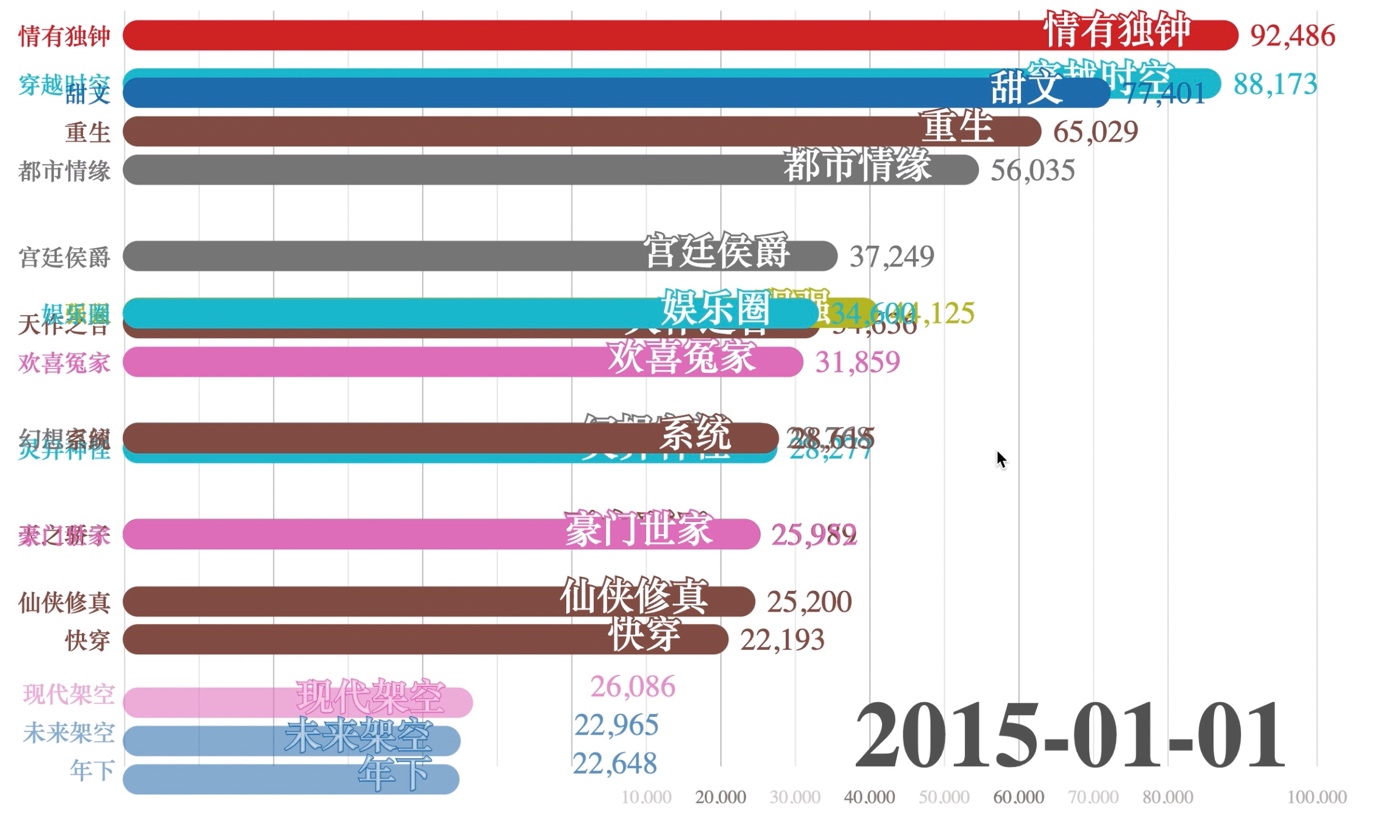


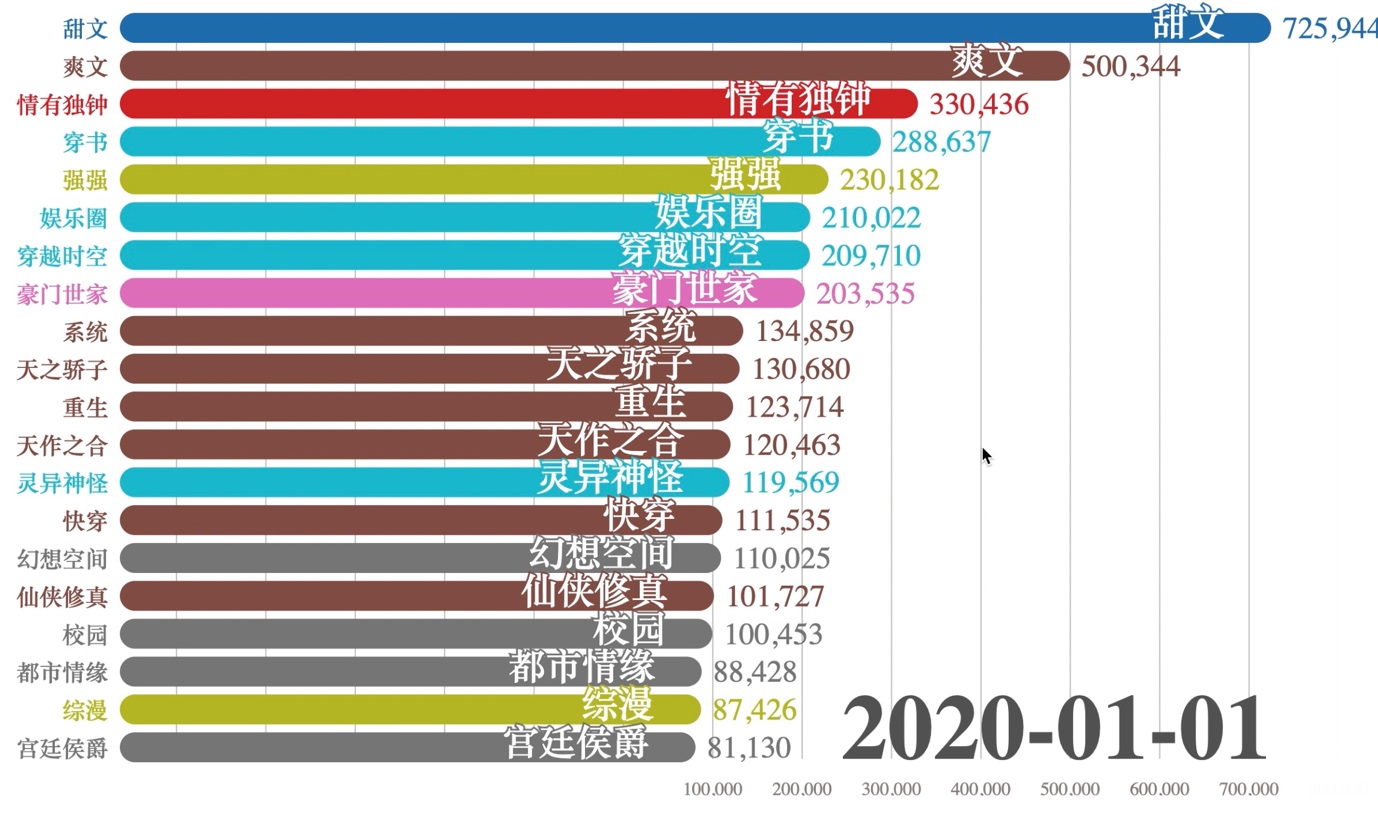
经过/anichart/toCsv.py中的文件格式与类型转换后，输出能够被anichart模块使用的tags\_by\_year.csv文件。toCsv.py中执行的主要任务包括处理json文件格式，去除多余标点、转换日期格式为yy-mm-dd，之后将数据转换为csv格式，并完成热度随年份累加的数据转换操作，最终输出文件格式如图：



文件输出成功后在浏览器打开html代码，选取输出文件，动态柱状图效果将在浏览器中展示，如下组图：

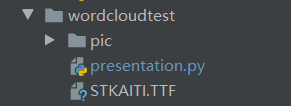






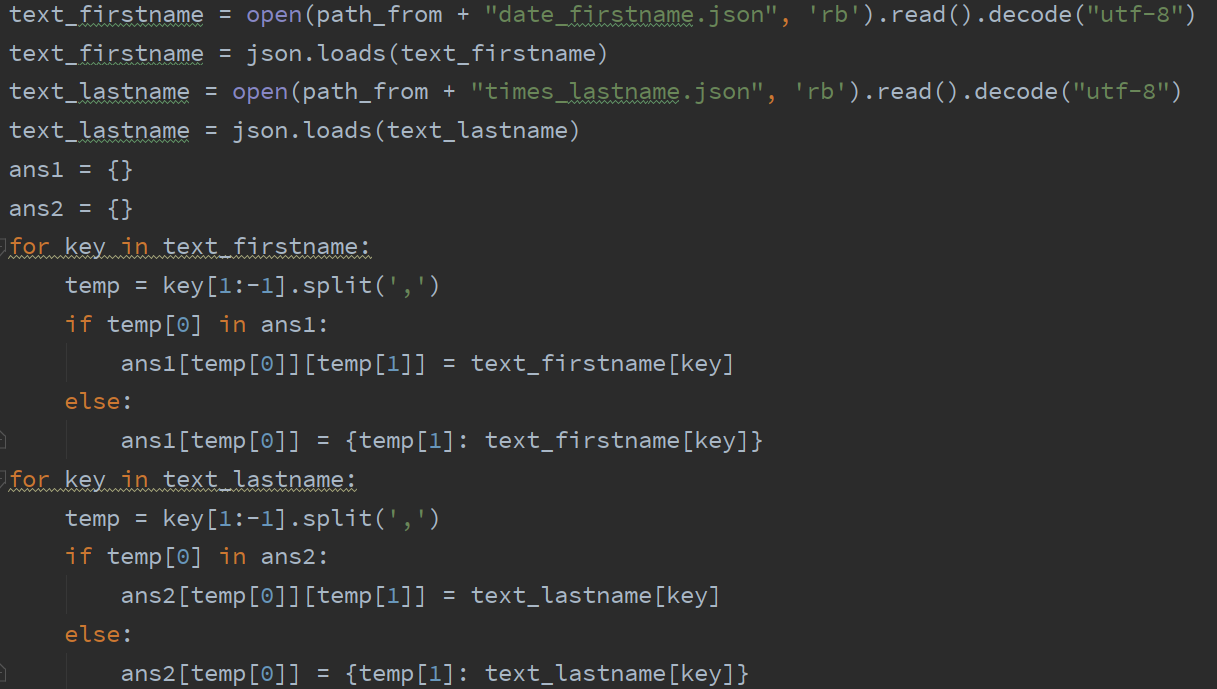
**词云——晋江百家姓**

词云部分的文件结构如下图，其中presentation.py是制作词云的主程序，STKAITI是打印词云所需要的字体部分。



词云结构图

presentation.py中，首先读取出姓和名的数据并进行相应的处理计算出使用频次，导出为字典属性留给后面生成词云图使用。



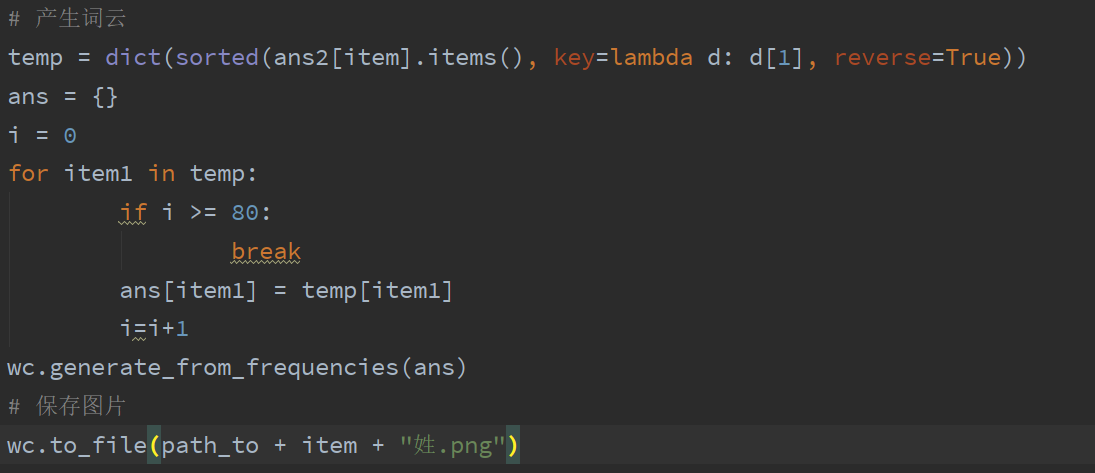
读取并处理数据

然后根据wordcloud库的要求，对词云图进行相应的属性设置，由于需要输出中文，所以需要将字体导入到项目内并在属性中设置好。



词云图属性设置

由于数据量过大，输出的词云图显得过于密集，在调试过后决定使用“每张图不超过80个字”的标准来对于词云图的数据进行要求：先按照频次进行倒序排序，然后取前80个键值对。最后按照频次生成词云图并且存在本地。姓和名使用同样的操作，只在保存图片的名称上略有区别。



输出词云图

文件保存之后直接在本地打开，如下组图：



名-2010 名-2011



名-2014 名-2015



名-2019 名-2020



姓-古色古香 姓-幻想未来



姓-架空历史 姓-近代现代

**遇到的困难及解决方案**

**数据获取的困难&解决方案**

数据获取的困难在于，晋江文学城网的游客访问存在限制，某些页面不登录就无法访问，从而自动化脚本就获取不到网页源码。

为了解决这个问题，本小组的解决方案如下：

1. 用自己的chrome浏览器登录晋江文学城，并在登录时勾选选项“保持登录状态一个月”
2. 将自己的chrome的用户设置复制到任意一个路径下（因为这个用户设置记录了登录状态）
3. 使用selenium包内的webdriver浏览器控制模块打开chrome浏览器，启动时用户设置的路径设置为2.中的路径。用webdriver访问晋江文学城
4. 如果未出现错误，此时打开的晋江文学城是已登录状态。这时代码就可以自动从webdriver中获取cookie从而创建会话。使用这个会话访问晋江文学城的子网站就不会有限制了

**流计算的困难&解决方案**

流处理的困难包括：

1. 由于从晋江爬取的源文件编码混乱，尤其是古早时期的数据爬取包含大量特殊字符、全角字符，在流计算前期的数据处理转存时我们碰到了很多麻烦。

为了解决这个问题，我们在流计算数据准备时使用了try except代码块，对不能识别的字符格式进行了抛弃处理。在进行所有数据的流计算时，为了避免未知编码错误，我们同样使用异常捕获和抛弃的方法，以数据量为支撑，获得计算的流畅性和效率。

1. 由于windows和linux操作系统上的区别，部分在win10上测试顺利的代码在集群上无法启动。例如在win10测试中，我们的getList方法使用了os包直接读取文件夹目录下的所有文件名；但是在linux上无法使用。

我们放弃了直接从文件夹内读取到文件名列表的方法，将getList修改为：将文件夹目录打印在控制台，再从控制台读取文件名列表。

1. 我们的代码文件有较为复杂的依赖调用关系，而我们只能往集群上提交一个主文件。我们尝试了将涉及的代码文件打包为zip，与主文件同时提交的方法，但import依赖关系处理较为困难、在集群上试错成本高，我们后来放弃了这种做法。

我们选择在集群上修改代码结构，从按层次划分改为按任务划分，将一次流处理涉及的方法转移到主文件里，直接提交了ast库作为唯一的依赖文件，成功解决了这个问题。