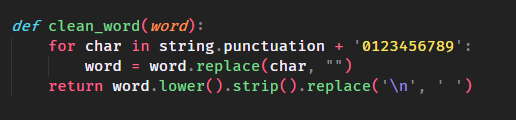
|  |
| --- |
| به‌نام خداوند بخشنده و مهربان  **تمرین سوم:** زمان اجرا **استاد:** محمد‌علی نعمت‌بخش  **درس:** تحلیل سیستم داده­های حجیم دستیاران: فاطمه‌ ابراهیمی، پریسا لطیفی، امیر سرتیپی  **نام و نام‌خانوادگی:** سید عمید اسدالهی مجد  **آدرس گیت:**<https://github.com/amidmajd/HW-3-hadoop-spark-comparison> |

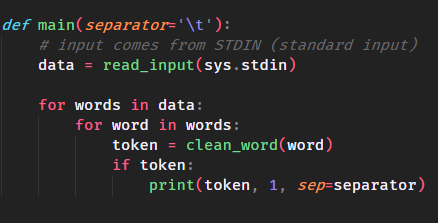
**عملیات Map و Reduce در Hadoop**

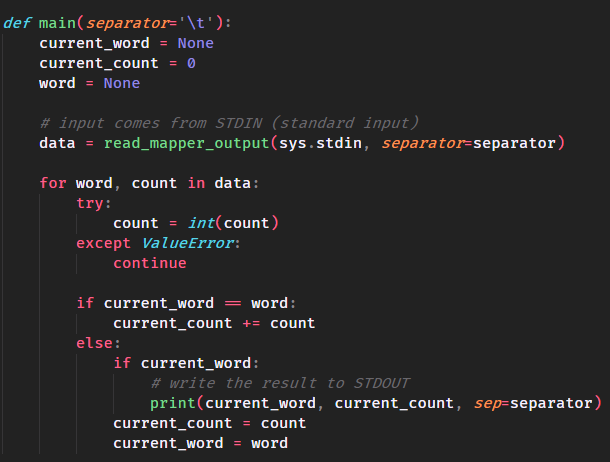
دو فایل **hadoop\_mapper.py** و **hadoop\_reducer.py** در پوشه src موجود است که مربوط به این عملیات می‌باشند. فایل ورودی متنی توسط Hadoop از طریق ورودی استاندارد (Standard Input) به هر بخش از عمیلات داده می‌شود و همچنین خروجی هر بخش نیز از خروجی استاندارد (Standard Output) توسط Hadoop دریافت می‌شود. این عمیات، با استفاده از قابلیت streaming در هدوپ انجام می‌شود که خروجی mapper را پس از مرتب‌سازی بر اساس کلمات، به عنوان ورودی به reducer می‌دهد. در هدوپ با بکارگیری streaming به صورت خودکار یک مرحله میانی تحت عنوان shuffle & sort روی خروجی نهایی مرحله map انجام می‌شود و سپس به عنوان ورودی به مرحله reduce داده می‌شود.

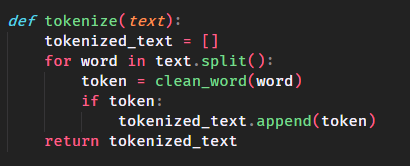
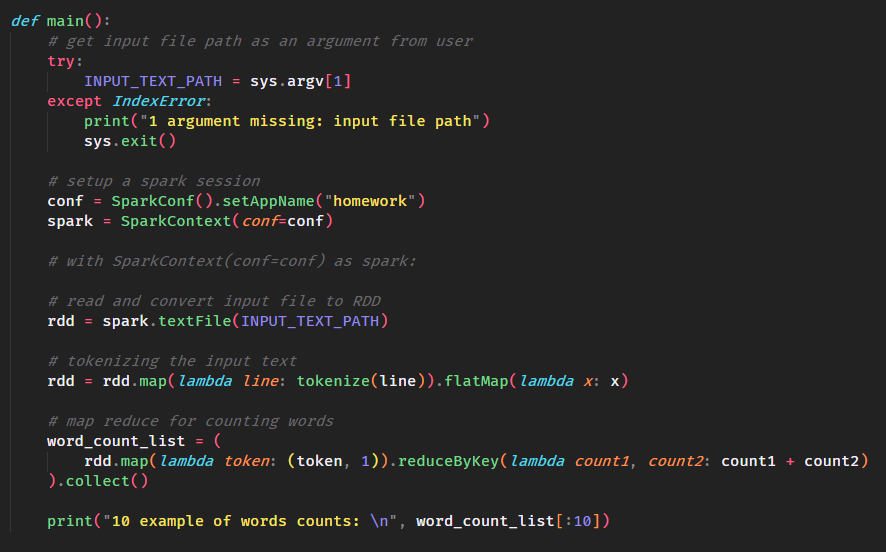
در این برنامه، دو تابع **read\_input** و **read\_mapper\_output** برای خواندن ورودی از تولید‌کننده‌های (Generators) پایتون استفاده شد که به‌جای اینکه ابتدا تمام عناصر را دریافت کنند و سپس یک لیست یا آرایه را برگردانند، به محض آماده شدن هر عنصر، آن را برمی‌گردانند. با استفاده از این روش به‌جای خواندن معمولی آرایه، می‌توان در مصرف حافظه صرفه‌جویی کرد و نیاز نیست تمام داده‌ها در حافظه بارگذاری شوند و سپس برگردانده شوند. همچنین در زمان اجرا نیز تاثیر بسزایی دارد زیرا نیاز نیست منتظر باشیم حلقه کامل انجام شود و سپس خروجی برگردانده شود و سپس توسط حلقه دیگری عملیات مورد نظر روی داده‌ها انجام شود.

تابع **clean\_word** وظیفه تمیز‌کردن هر کلمه را دارد. این تابع علائم سجاوندی (مانند نقطه، ویرگول، پرانتز و یا علامت نقل قول) را حذف می‌کند و کلمه را به حالت حروف کوچک (lowercase) تبدیل می‌کند.







**عملیات Map و Reduce در Spark**

فایل spark\_map\_reduce.py عمیات نگاشت-کاهش را در اسپارک پیاده‌سازی می‌کند. در این فایل تابع tokenize وظیفه شکستن متن ورودی خود به توکن‌ها را برعهده دارد. در این بخش از برنامه پس از خواندن فایل متنی و تبدیل آن به یک RDD، با استفاده از تابع map اسپارک، ابتدا هر جمله‌ی درون RDD را به توکن‌های مختلف آن‌ تبدیل می‌کنیم و سپس با استفاده از تابع flatMap اسپارک، تمام این لیست‌های حاوی توکن را ادغام کرده و RDD را تبدیل به لیستی از توکن‌ها می‌نماییم. در انتها با استفاده از تابع map اسپارک، کلمات (توکن‌ها) را تبدیل به دوتایی‌های حاوی کلمه و عدد یک می‌نماییم (عمل نگاشت) و سپس با استفاده از تابع reduceByKey اسپارک، عملیات کاهش را پیاده‌سازی کرده و تعداد هر کلمه را بدست می‌آوریم.

مقایسه زمان اجرا در هدوپ و اسپارک

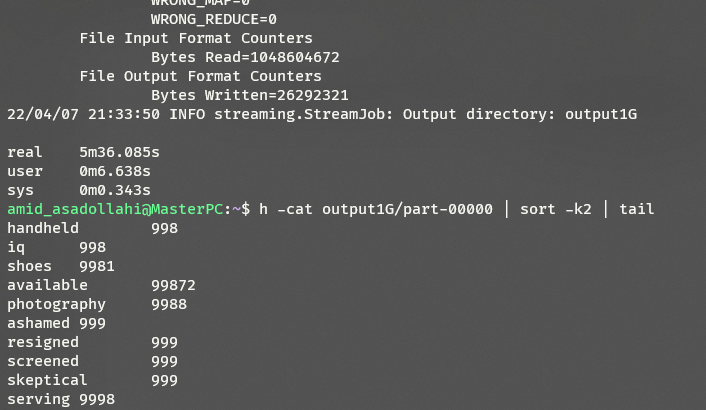
برای اجرای برنامه شمارش کلمه با استفاده از هدوپ و بررسی زمان اجرا از دستور زیر استفاده شد:

**time** **hadoop** **jar** $HADOOP\_HOME/share/hadoop/tools/lib/hadoop-streaming-2.10.1.jar   
**-files** /home/amid\_asadollahi/src/hadoop\_mapper.py,/home/amid\_asadollahi/src/hadoop\_reducer.py   
**-mapper** hadoop\_mapper.py **-reducer** hadoop\_reducer.py **-input** file10G.txt **-output** output10G

برای اجرای برنامه شمارش کلمه با استفاده از اسپارک و بررسی زمان اجرا از دستور زیر استفاده شد:

**time** **spark-submit** **--master** yarn **--deploy-mode** **client** ~/src/spark\_map\_reduce.py file10G.txt output1G.txt

این دستورات با استفاده از هریک از فایل‌های ۱، ۵، ۱۰ و ۱۲ گیگابایتی اجرا شدند.



خروجی دستور yarn application -status APP\_ID

