

Chameleon Cloud Tutorial

Implementation of MapReduce Program (Python)



**Program Solicitation # NSF 13-602**

CISE Research Infrastructure:

Mid-Scale Infrastructure - NSFCloud (CRI: NSFCloud)

**Hadoop - Implementation of Map Reduce (Python) Objective**

In this tutorial, we will discuss about the Map and Reduce program, its implementation.

|  |  |
| --- | --- |
| Tutorial Action | Time Required |
| **Step 1: Implementation of multiple MapReduce programs**  You will learn how to implement MapReduce programs on everyday scenario’s | 15 minutes |
| **Step 2: Execution of MapReduce programs**  In this session you will get familiar on how to execute the MapReduce program and the analysis of the input data. | 10 minutes |

**Prerequisites**

The following prerequisites are expected for successful completion of this tutorial:

 Chameleon Cloud account [(http://chameleoncloud.org/user/register/](http://chameleoncloud.org/user/register/))

 SSH client (Windows users: download PuTTY from here:

[http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)

 A basic knowledge of Linux.

 Installation of Hadoop and Map reduce.

 Single node set up.

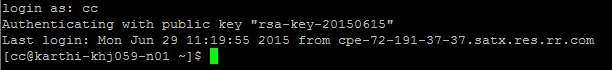
**Step 1: Implementation of MapReduce**

 Login to the system with user id ‘CC’ and enter to establish the connection.

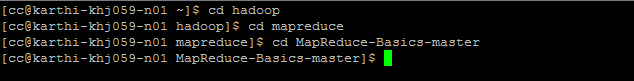
Login as ‘CC’



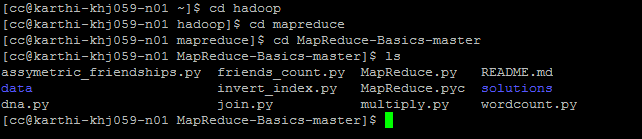
Enter to establish connection (Authenticate with the public key)



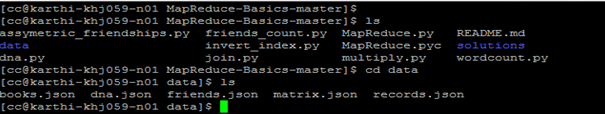
 Change the directory to MapReduce-Basics-master ( hadoop-mapreduce-MaReduce-Basics-master)



* View the files present in the MapReduce-Basics-master directory using ls command.



* Move to the data directory where the input data is stored.



* **Inverted Index:**

In this the input Json file contains document \_id and the sample text, where

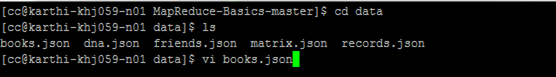
**document\_id** - An document identifier formatted as string

**text**- Document formatted as a string.

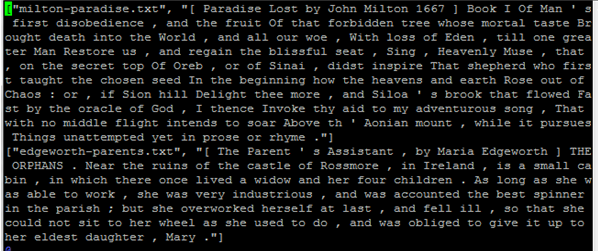
Output file contains (word, document ID list) where word is an string and the document ID list is list of strings.

**Input text file:**

Open the data directory to view the input json file. Use the below command shown in the screen print to view the books.json file.



Below books.json file is displayed after executing the above commands shown in the screen print.



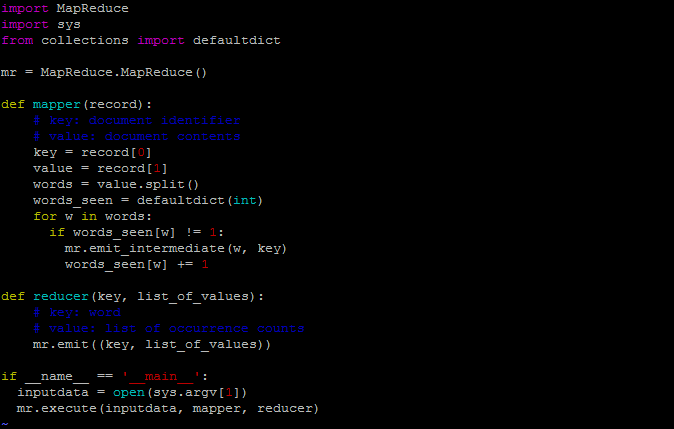
Open invert\_index.py file to view the map and reducer program which is used to generate the output file.

Defaultdict: whenever you need a dictionary, and each element’s value should start with a default value, so use a **defaultdict**.

**Mapper:** Document identifier content is stored in key and the document content is stored to value.

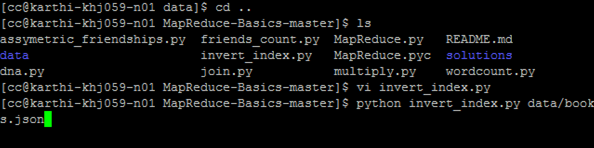
It uses for loop for the words and if loop to match the words from the different txt files.

**Reducer:** It lists the word and the txt file which contains the key word.

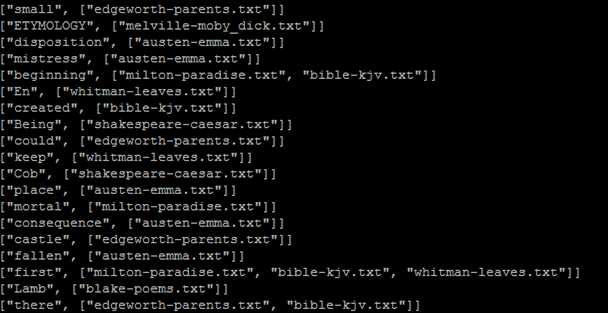


**Output of the Invert\_index file:**

To generate the output file we need to execute the below command



**Output file:** The output file displays the word and its corresponding text file.



* **Friends Count:**

In this program each key is a person and each value is a friend of the person. This program mainly counts number of friends each person has.



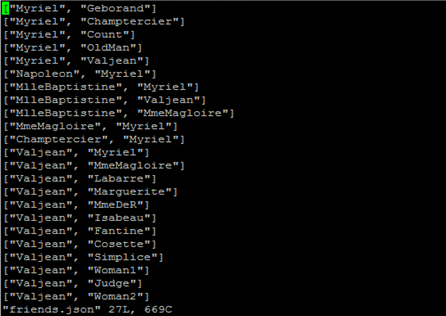
The input is a two element list: [personA, personB] where

**personA**- Name of a person formatted as a string

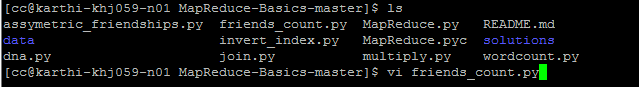
**personB**- Name of the personA’s friend formatted as a string.

The output of the program should be (person, friend count) where person is a string and the friend count is an integer describing number of friends that the person has.

**Input file( friends.json):** Input file contains the list of friends names.

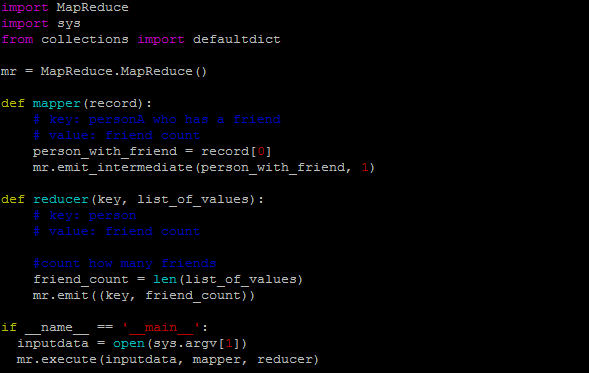


Open friends\_count.py file to view the code used to get the count of the friends.



**Mapper:** Key stores value of the personA’s name who has friend and value stores the friends count(PersonA).

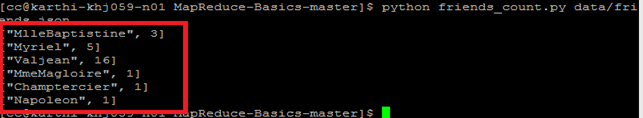
**Reducer:** mr.emit displays the personA name and the count of friends personA has.



**Output:** Execute the below command to display the results



Output file displays the person name and the count of friends.



* **Asymmetric Friendship:**

In this program we implemented MapReduce algorithm to generate a list of all non symmetric friend relationship.

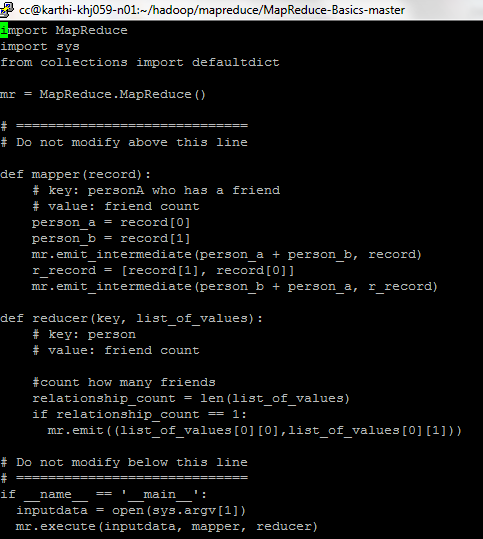
The input is a two element list: [personA, personB] where

**personA**- Name of a person formatted as a string

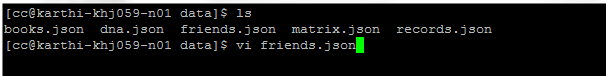
**personB**- Name of the personA’s friend formatted as a string.

****

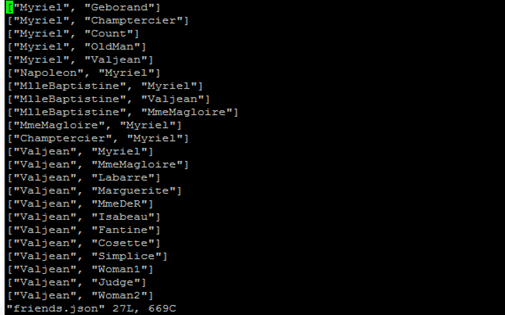
**Mapper:** record[0] value is stored to person\_a and record[1] value is stored to person\_b where the final output of the mapper displays both (friend\_a, friend\_b) list along with(friend\_b, friend\_a) list.



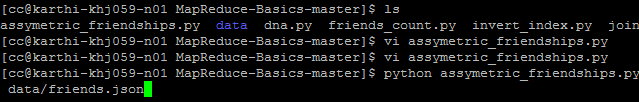
Open friends.json file to view the input folder, where the json file displays the list of friends.



By executing above command it displays the list of friends data which is used to execute the specified algorithm.



Execute the program by giving the below command by providing file which contain the program and the input data file which is used for processing.



After executing the above program it displays the asymmetric relation values both (friend\_a, friend\_b) and (friend\_b , friend\_a).

