**Practical 4**

**Aim: Write an Hadoop MapReduce Program in Python**

**🡪**

## Create the mapper.py

*#!/usr/bin/env python*

"""mapper.py"""

*import* sys

*# input comes from STDIN (standard input)*

*for* line in sys.stdin:

*# remove leading and trailing whitespace*

    line = line.strip()

*# split the line into words*

    words = line.split()

*# increase counters*

*for* word in words:

*# write the results to STDOUT (standard output);*

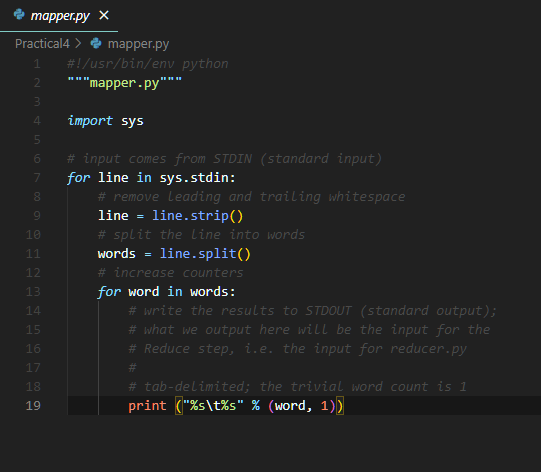
*# what we output here will be the input for the*

*# Reduce step, i.e. the input for reducer.py*

*#*

*# tab-delimited; the trivial word count is 1*

        print ("%s\t%s" % (word, 1))



## Create the reducer.py

*#!/usr/bin/env python*

"""reducer.py"""

*from* operator *import* itemgetter

*import* sys

current\_word = None

current\_count = 0

word = None

*# input comes from STDIN*

*for* line in sys.stdin:

*# remove leading and trailing whitespace*

    line = line.strip()

*# parse the input we got from mapper.py*

    word, count = line.split('\t', 1)

*# convert count (currently a string) to int*

*try*:

        count = int(count)

*except* ValueError:

*# count was not a number, so silently*

*# ignore/discard this line*

*continue*

*# this IF-switch only works because Hadoop sorts map output*

*# by key (here: word) before it is passed to the reducer*

*if* current\_word == word:

        current\_count += count

*else*:

*if* current\_word:

*# write result to STDOUT*

            print ("%s\t%s" % (current\_word, current\_count))

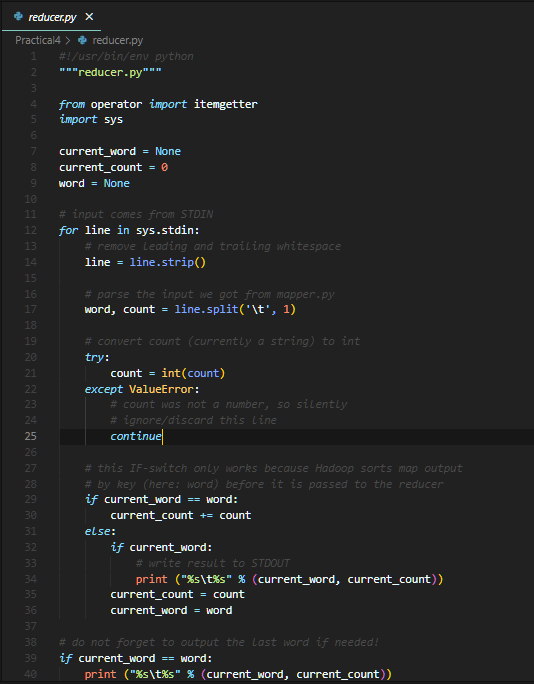
        current\_count = count

        current\_word = word

*# do not forget to output the last word if needed!*

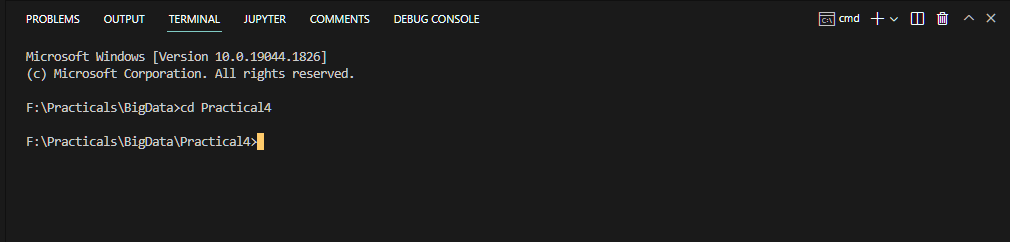
*if* current\_word == word:

    print ("%s\t%s" % (current\_word, current\_count))

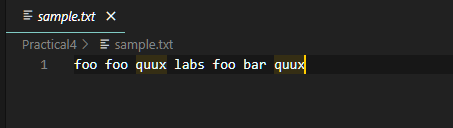


## Running mapper and reducer without Hadoop HDFS

### Step 1: Open **Command Prompt** where the mapper.py and reducer.py is located



### Step 2: To Execute the program create one file in the same location with name sample.txt



### Step 3: Now run the following command to get the output

type .\sample.txt | python .\mapper.py | sort |python .\reducer.py

## Running the Python Code on Hadoop

### Step 1:Download example input data

We will use three eBooks from Project Gutenberg for this example:

* [The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson](http://www.gutenberg.org/etext/20417)
* [The Notebooks of Leonardo Da Vinci](http://www.gutenberg.org/etext/5000)
* [Ulysses by James Joyce](http://www.gutenberg.org/etext/4300)

Download each eBook as text files in Plain Text UTF-8 encoding and store the files in a local temporary directory of choice.

## Copy local example data to HDFS

Before we run the actual MapReduce job, we must first copy the files from our local file system to Hadoop’s HDFS.

### Step 1: Open Command Prompt in Administration Mode and change the present working directory to the **C:\Hadoop\hadoop-3.3.3\sbin**

### Step 2: Now run the command **.\start-all.cmd**

### Step3: Now change the present working directory to **C:\Hadoop\hadoop-3.3.3\bin** and run the command

#### hadoop dfs –copyFromLocal ‘path of the downloaded sample file’ ‘path to store on the hdfs’

hadoop dfs -copyFromLocal "F:\Practicals\BigData\Practical4\Data"  hdfs://localhost:9000/Harsh

hadoop dfs -ls /Harsh

### Step 4: To check the files are uploaded to the Hadoop HDFS the visit <http://localhost:9870/dfshealth.html#tab-overview> 🡪 go to utilities in the navigation bar and click on the Browse the file system

### Step 5: Run the MapReduce job

hadoop jar C:\Hadoop\hadoop-3.3.3\share\hadoop\tools\lib\hadoop-streaming-3.3.3.jar -file F:\Practicals\BigData\Practical4\mapper.py -mapper "python mapper.py" -file F:\Practicals\BigData\Practical4\reducer.py -reducer "python reducer.py" -input hdfs://localhost:9000/Harsh/sample.txt -output /output

### Step 6: Check if the result is successfully stored in HDFS directory /output

hadoop dfs -ls /output

### Step 7: To check the output is generated to the Hadoop HDFS the visit <http://localhost:9870/dfshealth.html#tab-overview> 🡪 go to utilities in the navigation bar and click on the Browse the file system

### Step 8: You can then inspect the contents of the file with the dfs -cat command:

hadoop fs -cat /output/part-00000