**PYTHON PRACTICE**

**To type** **Actions & Output**

1. Su – main user Change user to main user
2. Sudo apt install python To Install Python
3. Su – hadoopuser To Change user……..… to ***- hadoopuser***
4. nano helloworld.py To Create file…………….Write **print “hellojoshi”**
5. python helloworld.py To Check the file ……….Output ***hellojoshi***
6. nano helloworld.java To create file in java ….Write

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

1. javac HelloWorld.java To Compile java
2. cat HelloWorld.java To See the content of the file, the same syntax above will display
3. cat HelloWorld.class Some computer language will be displayed, not understandable
4. java HelloWorld Will show ***Hello, World***
5. python Will change command to python, prompted by ***>>>***
6. >>> print "hellojoshi" ***hellojoshi***will be displayed
7. >>> 2+3 Hit ENTR .….. ***5*** will be displayed
8. >>> 100/3 Hit ENTR ….…***33*** will be displayed
9. >>> x=100/3 Hit ENTR
10. >>> x Hit ENTR …… ***33*** will be displayed
11. >>> y=x/3 Hit ENTR
12. >>> y Hit ENTR …… ***11*** will be displayed
13. >>> y+3 Hit ENTR ……. ***14*** will be displayed
14. >>> exit() to exit from python and go to hadoop user
15. nano mypython.py To Create file

x = 100/3

x

x+5

x= x+5

x

x=x/3

x

1. print x
2. python mypython.py Hit ENTR to get the value of x, result will be displayed
3. python To enter python, terminal will be prompted by >>>
4. >>> x = "Hello World" Hit ENTR
5. >>> print x Hit ENTR …..***Hello World*** will be displayed
6. >>> x.split() Will split words like ***[ ‘Hello’, ‘World’ ]***
7. >>> y = "Hello World" Hit ENTR
8. >>> print y Hit ENTR …..***Hello World*** will be displayed
9. >> b = y.split() Hit ENTR
10. >> print b Will split words like ***[ ‘Hello’, ‘World’ ]***
11. >> b[0] Will display ***‘Hello’***
12. >> b[1] Will display ***‘World’***
13. >> z = " Hello World " Hit ENTR

34. >> z will show ‘ Hello World ‘

35. >> z.strip() Will remove unwanted space from before and after ***‘Hello World ‘***

1. >> exit() to exit from python and go to hadoop user
2. nano forloopexp.py To create file in python ….. write

m = ['carrot', 'tomatoe', 'muli' ]

for n in m:

print(n)

1. python forloopexp.py Will show ….. ***carrot***

***tomato***

***muli***

1. nano forloopexp.py To create file in python ….. write

m = ['carrot', 'tomatoe', 'muli' ]

for n in m:

print "I am going to buy"+n

1. python forloopexp.py Will show ….. ***I am going to buycarrot***

***I am going to buytomato***

***I am going to buymuli***

1. nano forloopexp.py To create file in python ….. write

m = ['carrot', 'tomatoe', 'muli', 'bindi', 'brookli', 'potetoe', 'karela' ]

for n in m:

print "I am going to buy"+n

1. python forloopexp.py Will show ….. ***I am going to buycarrot***

***I am going to buytomato***

***I am going to buymuli***

***I am going to buybindi***

***I am going to buybrookli***

***I am going to buypotatoe***

***I am going to buykarela***

1. python To enter python, terminal will be prompted by >>>
2. >>> p='samosa' Hit ENTR
3. >>> print p Will display …… ***samosa***
4. >>> p == 'samosa' Will display …… ***true***
5. >>> p == '"kachori' Will display …… ***false***
6. >> exit() to exit from python and go to hadoop user
7. nano forloopexp.py To create file in python ….. write

m = ['carrot', 'tomatoe', 'muli', 'bindi', 'brookli', 'potetoe', 'karela', ‘samosa’ ]

for n in m:

print "I am going to buy"+n

1. python forloopexp.py Will show ….. ***I am going to buycarrot***

***I am going to buytomato***

***I am going to buymuli***

***I am going to buybindi***

***I am going to buybrookli***

***I am going to buypotatoe***

***I am going to buykarela***

***I am going to buysamosa***

1. nano forloopexp.py To create file in python ….. write

m = ['carrot', 'tomatoe', 'muli', 'bindi', 'brookli', 'potetoe', 'karela', ‘samosa’ ]

for n in m:

if n == 'samosa' :

print 'good'

else:

print 'bad'

1. python forloopexp.py Will show ….. ***bad***

***bad***

***bad***

***bad***

***bad***

***bad***

***bad***

***good***

***Link for practicing map reduce in python***

<https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>

1. nano mapper.py create file with the below syntax for mapping write……

*#!/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)

1. nano reducer.py create file with the below syntax for reducing write…….

*#!/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)

1. echo "foo foo quux labs foo bar quux" | python mapper.py Get display as below

foo 1

foo 1

quax 1

lab 1

foo 1

bar 1

quas 1

1. echo "foo foo quux labs foo bar quux" | python mapper.py | sort -k1,1 | python reducer.py

displays

bar 1

foo 3

lab 1

quax 2

1. wget <http://www.gutenberg.org/files/5000/5000-8.txt> Upload file 5000-8.txt
2. wget <http://www.gutenberg.org/files/4300/4300-0.txt> Upload file 4300-8.txt
3. wget <http://www.gutenberg.org/cache/epub/20417/pg20417.txt> Upload file pg20417.txt
4. mkdir gutenberg Create directory gutenberg
5. mv 5000-8.txt gutenberg/ Move file 5000-8.txt to gutenberg
6. mv 4300-0.txt gutenberg/ Move file 4300-8.txt to gutenberg
7. mv pg20417.txt gutenberg/ Move file pg20417.txt to gutenberg
8. cd gutenberg To go directory gutenberg
9. ls To view the 3 above files moved or not
10. jps To check all 6 services are available if not…

if not go for step 65 and 66 and check again jps

1. start-dfs.sh
2. start-yarn.sh
3. hadoop fs -mkdir /user/hduser/ To create hadoop file and directory
4. hadoop fs -mkdir /user/hduser/Gutenberg To create a file named Gutenberg
5. hadoop dfs -copyFromLocal gutenberg /user/hduser/Gutenberg To copy files in HDFS

**To make jar file write the following command in terminal** (4 space before mapper and 3 space before reducer)

1. hadoop jar /usr/local/hadoop-2.8.4/share/hadoop/tools/lib/hadoop-\*streaming\*.jar -file mapper.py -mapper mapper.py -file reducer.py -reducer reducer.py -input /user/hduser/gutenberg/\* -output /user/hduser/gutenberg-output5

**To get the output write the following command in the terminal** (Can be guided from HDFS browser)

1. hadoop fs -cat /user/hduser/gutenberg-output5/part-00000

m = ['carrot', 'tomatoe', 'muli' ]