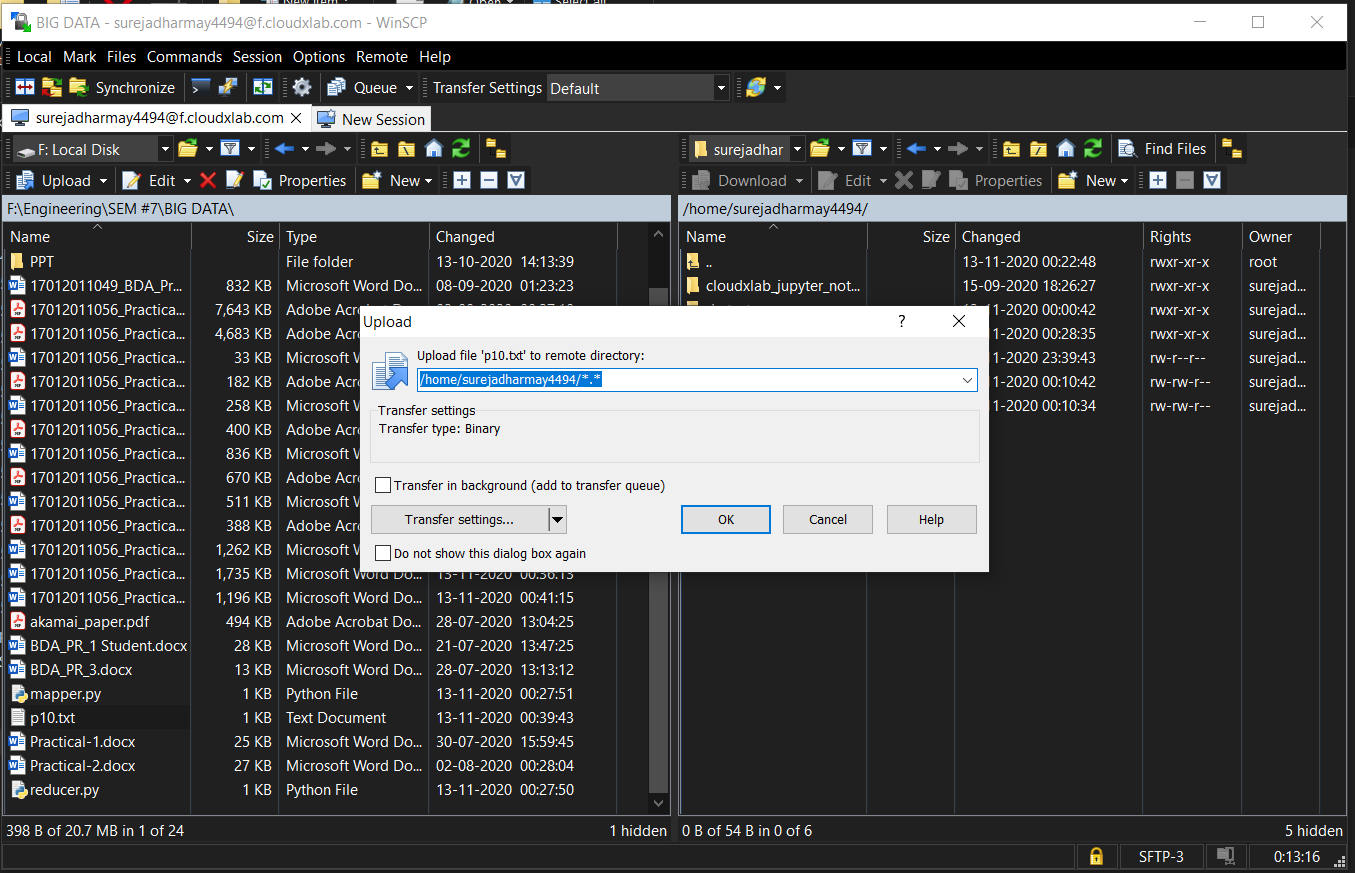
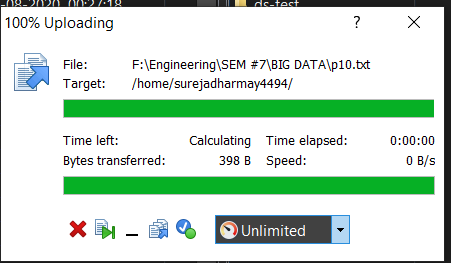
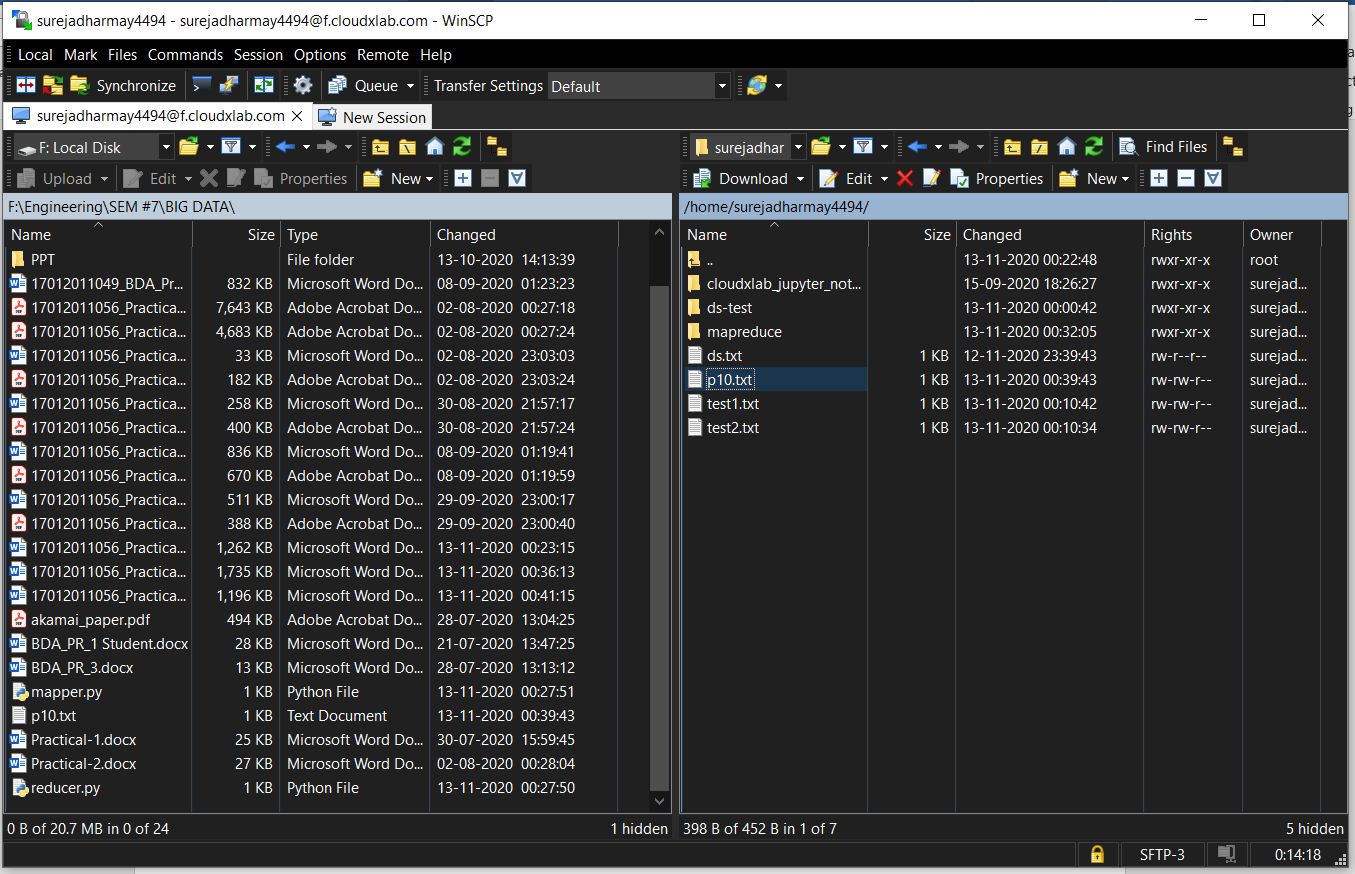
Practical-10

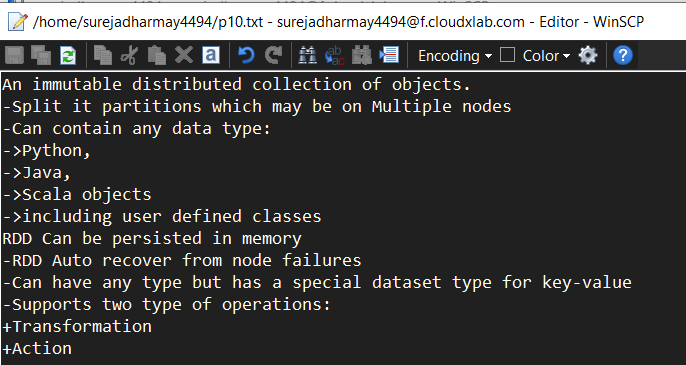
|  |  |
| --- | --- |
| **TASK :** | Create and upload one text file with some data in it. Remember the number of lines in it must be a minimum of 10. Now read this file using the Scala and Python interface in cloudxlab and exit both the interface. |



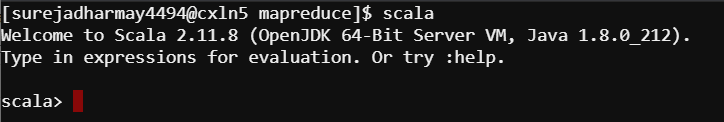


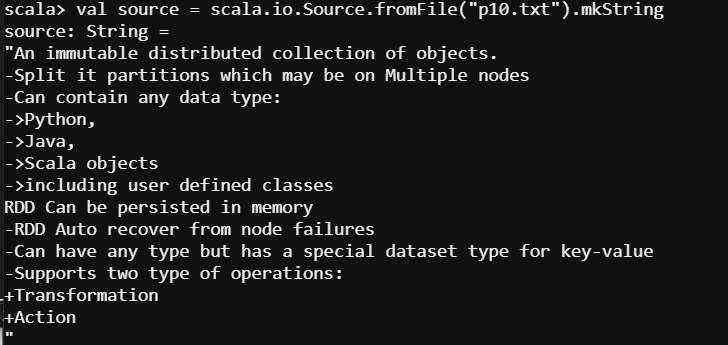


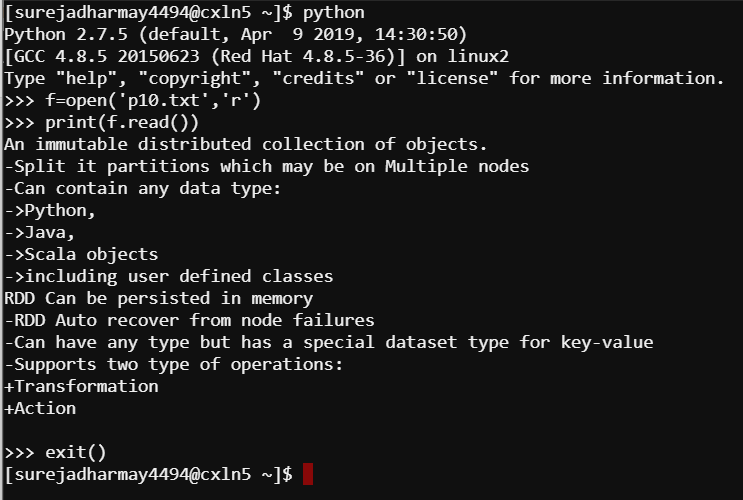
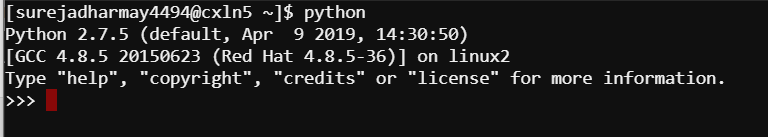
**P10-sample.txt**

****

**Scala :**

****

****

**Python :**

**TASK : Open Jupyter Notebook in CloudXLab and initialize Python interface in it and**

**Perform Apache Spark reading file and check the version of the spark.**

**Code :**

import os

import sys

os.environ["SPARK\_HOME"] = "/usr/spark2.4.3"

os.environ["PYLIB"] = os.environ["SPARK\_HOME"] + "/python/lib"

# In below two lines, use /usr/bin/python2.7 if you want to use Python 2

os.environ["PYSPARK\_PYTHON"] = "/usr/local/anaconda/bin/python"

os.environ["PYSPARK\_DRIVER\_PYTHON"] = "/usr/local/anaconda/bin/python"

sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")

sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

#Now, initialize the entry points of Spark: SparkContext and SparkConf

sc.stop()

from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName("appName")

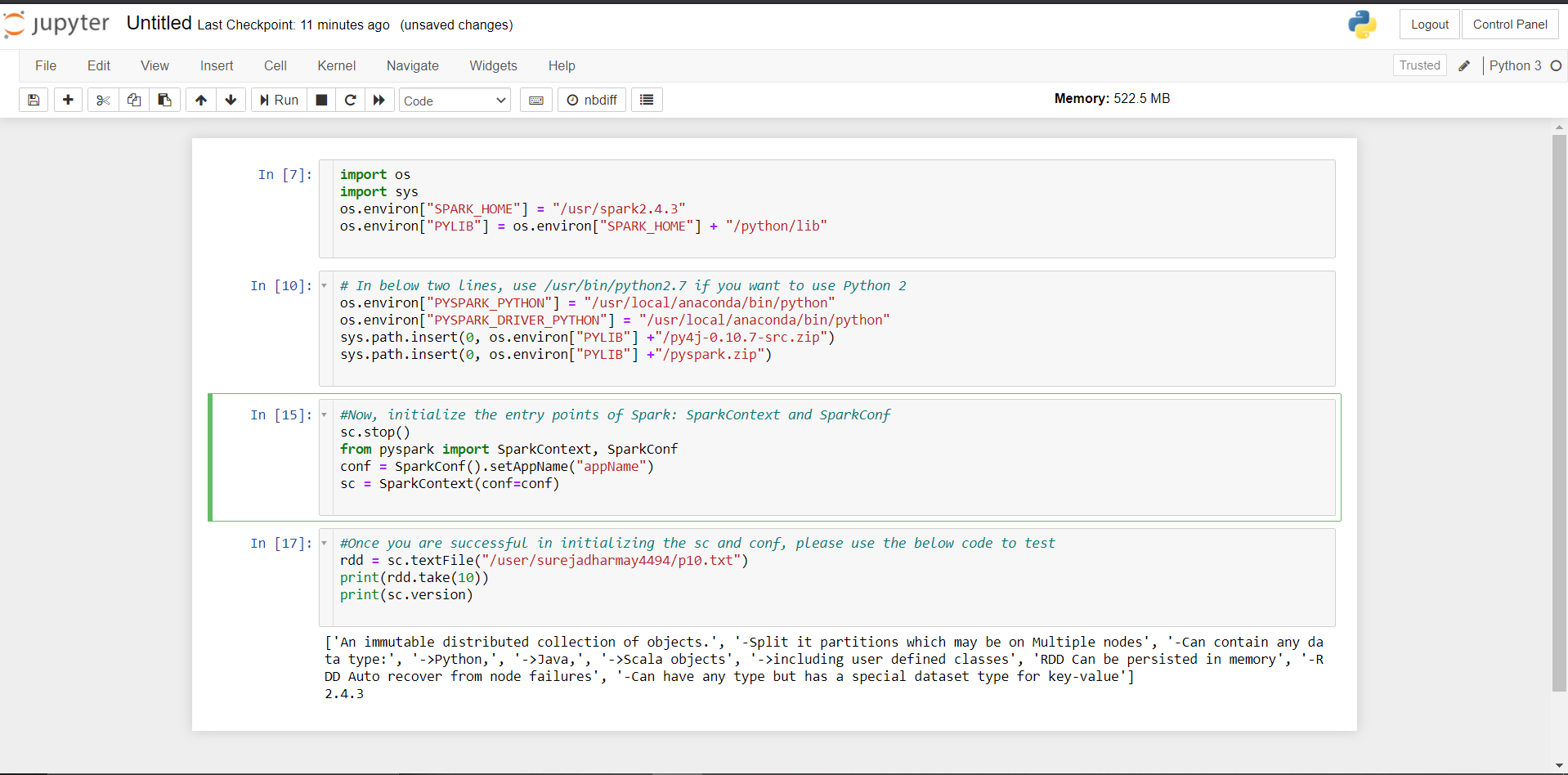
sc = SparkContext. setAppName(conf=conf)

#Once you are successful in initializing the sc and conf, please use the below code to test

rdd = sc.textFile("/user/surejadharmay4494/p10.txt")

print(rdd.take(10))

print(sc.version)

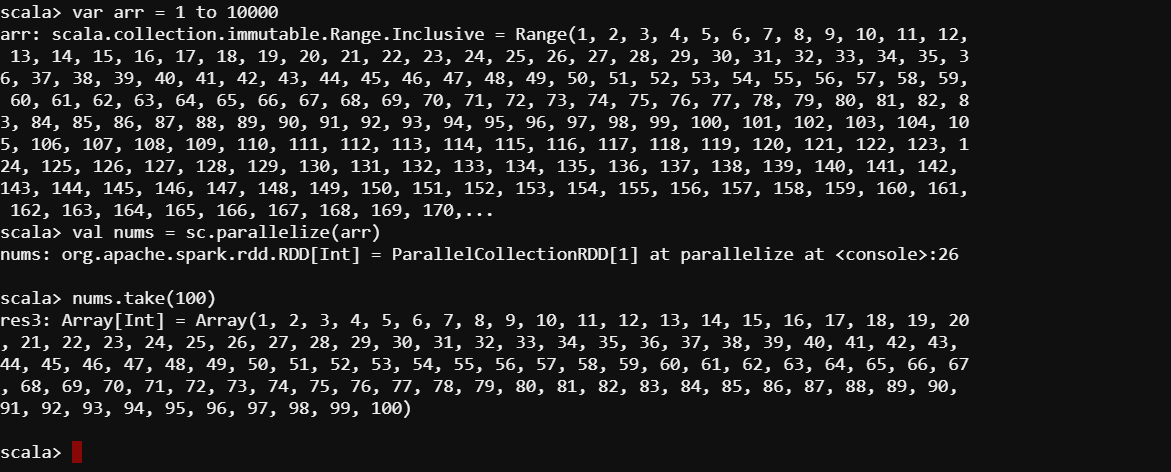
****

****

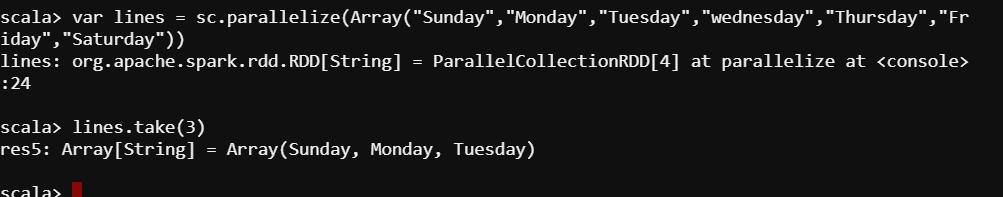
**TASK : Create RDD using by distributing existing object ( Using parallelized collection ) in**

**SCALA Interface .**

* Create an array of number 1 to 10000 and parellize it and take it first 100 numbers.



* Create an array of weekdays and take the first 3 days of the week.



**TASK : Use the Code and run in Python Interface ( Using Jupyter Notebook ) to find Average**

**Number of friends using the data given in and map the code understanding explained in the**

**lab and lectures**

**Code :**

import os

import sys

os.environ["SPARK\_HOME"] = "/usr/spark2.4.3"

os.environ["PYLIB"] = os.environ["SPARK\_HOME"] + "/python/lib"

# In below two lines, use /usr/bin/python2.7 if you want to use Python 2

os.environ["PYSPARK\_PYTHON"] = "/usr/local/anaconda/bin/python"

os.environ["PYSPARK\_DRIVER\_PYTHON"] = "/usr/local/anaconda/bin/python"

sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")

sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

#Now, initialize the entry points of Spark: SparkContext and SparkConf

from pyspark import SparkContext, SparkConf

conf = SparkConf().setMaster("local").setAppName("FriendsByAge")

sc = SparkContext(conf = conf)

def parseLine(line):

fields = line.split(',')

age = int(fields[2])

numFriends = int(fields[3])

return (age, numFriends)

lines = sc.textFile("/user/ surejadharmay4494/4lecref\_friends.csv")

rdd = lines.map(parseLine)

totalsByAge = rdd.mapValues(lambda x: (x, 1)).reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1]))

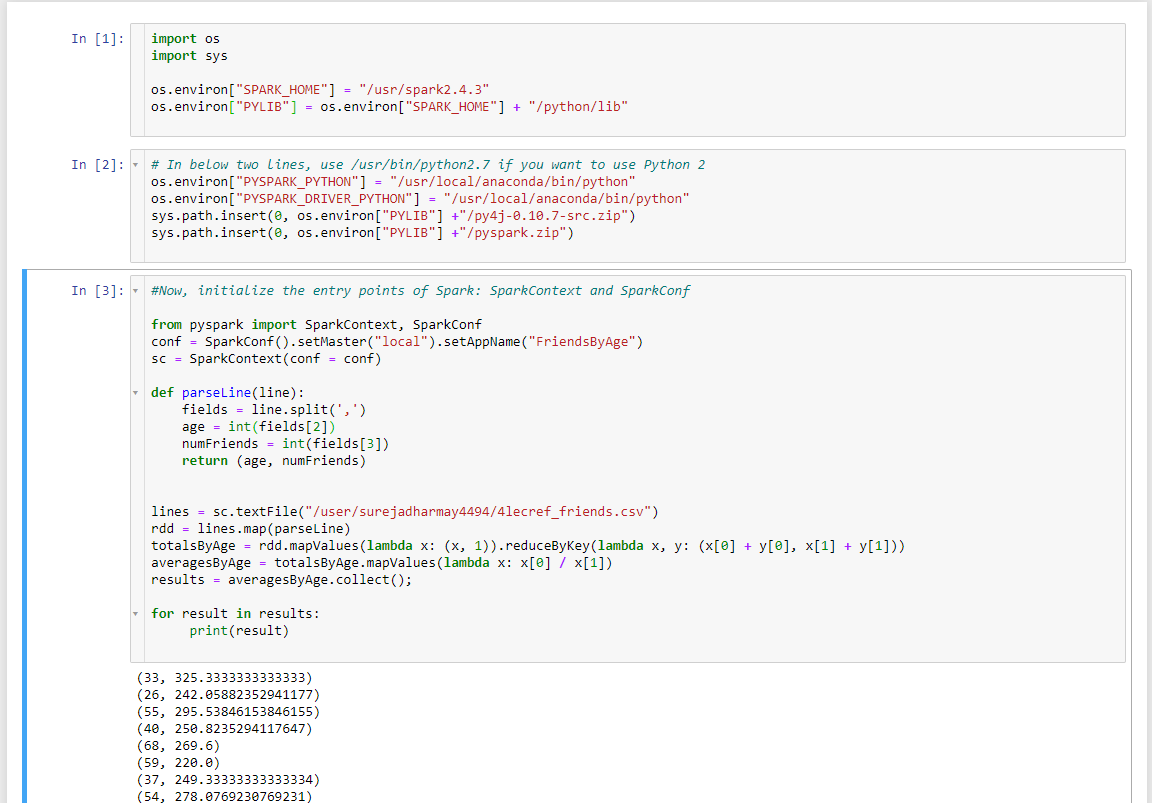
averagesByAge = totalsByAge.mapValues(lambda x: x[0] / x[1])

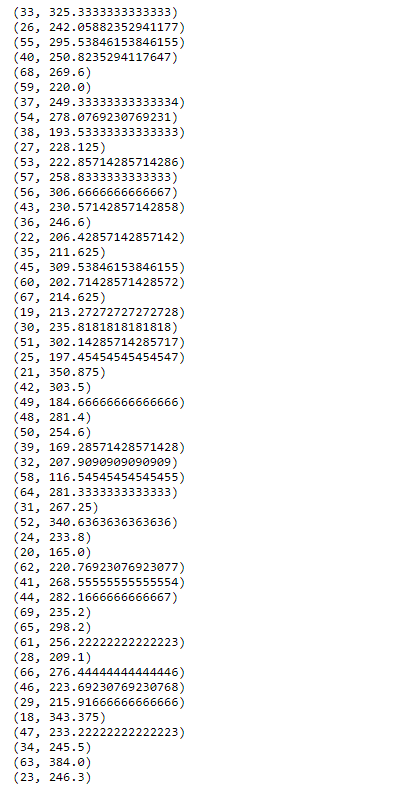
results = averagesByAge.collect();

for result in results:

print(result)

**Output :**



****

**TASK : Use the dataset given and check the code to find the minimum temperature by the**

**Location and understand it and modify it for to find the maximum temperature by the**

**Locations .**

**Code :**

import os

import sys

os.environ["SPARK\_HOME"] = "/usr/spark2.4.3"

os.environ["PYLIB"] = os.environ["SPARK\_HOME"] + "/python/lib"

# In below two lines, use /usr/bin/python2.7 if you want to use Python 2

os.environ["PYSPARK\_PYTHON"] = "/usr/local/anaconda/bin/python"

os.environ["PYSPARK\_DRIVER\_PYTHON"] = "/usr/local/anaconda/bin/python"

sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")

sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

#Now, initialize the entry points of Spark: SparkContext and SparkConf

from pyspark import SparkContext, SparkConf

sc.stop()

conf = SparkConf().setMaster("local").setAppName("MinTemperatures")

sc = SparkContext(conf = conf)

def parseLine(line):

fields = line.split(',')

stationID = fields[0]

entryType = fields[2]

temperature = float(fields[3]) \* 0.1 \* (9.0 / 5.0) + 32.0

return (stationID, entryType, temperature)

lines = sc.textFile("/user/ surejadharmay44945temperatures.csv")

parsedLines = lines.map(parseLine)

minTemps = parsedLines.filter(lambda x: "TMIN" in x[1])

stationTemps = minTemps.map(lambda x: (x[0], x[2]))

minTemps = stationTemps.reduceByKey(lambda x, y: min(x,y))

results = minTemps.collect();

for result in results:

print(result[0] + "\t{:.2f}F".format(result[1]))

**Output :**

****

**TASK : Use the given dataset of customers and their spend and find how much amount is spent**

**By the individual customer total creating proper RDD in the spark using Python .**

**Code :**

import os

import sys

os.environ["SPARK\_HOME"] = "/usr/spark2.4.3"

os.environ["PYLIB"] = os.environ["SPARK\_HOME"] + "/python/lib"

# In below two lines, use /usr/bin/python2.7 if you want to use Python 2

os.environ["PYSPARK\_PYTHON"] = "/usr/local/anaconda/bin/python"

os.environ["PYSPARK\_DRIVER\_PYTHON"] = "/usr/local/anaconda/bin/python"

sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")

sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

#Now, initialize the entry points of Spark: SparkContext and SparkConf

sc.stop()

from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName("appName")

sc = SparkContext(conf=conf)

#Once you are successful in initializing the sc and conf, please use the below code to test

def extractCustomerPricePairs(line):

fields = line.split(',')

return (int(fields[0]), float(fields[2]))

rdd = sc.textFile("hdfs:///user/surejadharmay4494/6customer-orders.csv")

mappedInput = rdd.map(extractCustomerPricePairs)

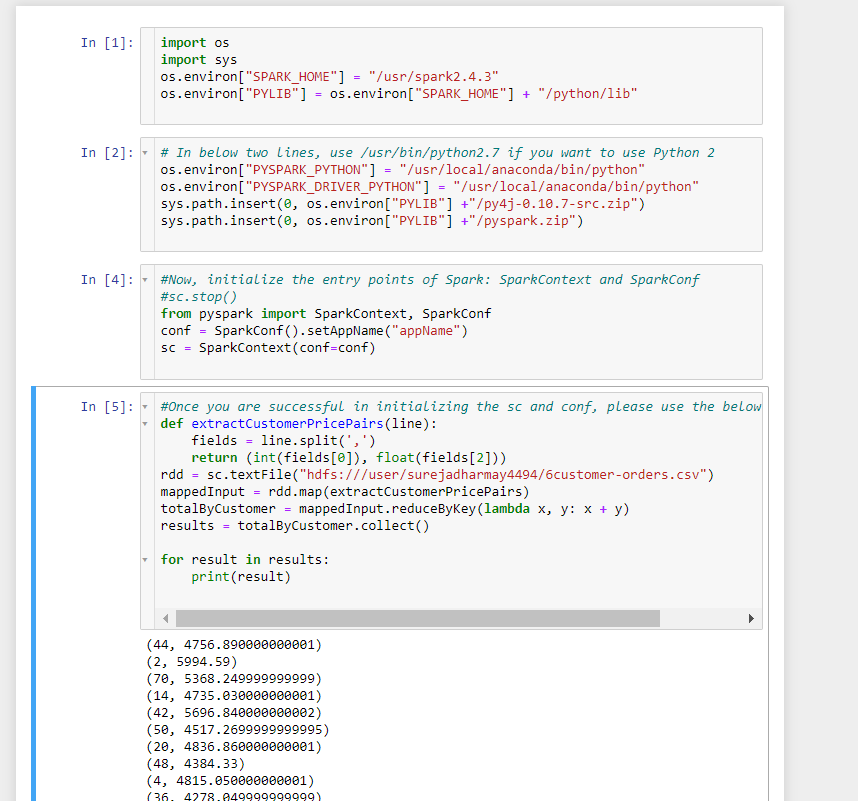
totalByCustomer = mappedInput.reduceByKey(lambda x, y: x + y)

results = totalByCustomer.collect()

for result in results:

print(result)

**Output:-**

****

