Big Data Management and Analytics Assignment 6
(a) Implement the word count example using PySpark

General 'pattern' of a (Py)Spark program:

```python
from pyspark import SparkConf, SparkContext

sparkConf = SparkConf()
    .setAppName("MyProgram")
    .setMaster("local")

sc = SparkContext(conf=sparkConf)
```

Configuration of a Spark application

- **Master URL to connect to local**: run locally with one thread
- **local[4]**: run locally with 4 cores

Represents connection to a Spark cluster. Can be used to create RDDs, accumulators and broadcast variables to that cluster.

*Note*: once a SparkConf object is passed to Spark, it is cloned and can no longer be modified by the user!
(a) Implement the word count example using PySpark

General ‘pattern’ of a (Py)Spark program:

```python
somevar = sc.parallelize(someData)
```

Creates from a provided iterable or collection an RDD.

- Dataset is cut into a certain number of partitions automatically
- # of partitions can be set manually by second parameter (someData, numberOfPartitions)
(a) Implement the word count example using PySpark

General 'pattern' of a (Py)Spark program:

```python
smth_mapped = somevar.map(f)

smth_reduced = smth_mapped.reduceByKey(
    lambda var1, var2 : var1 op var2)

smth_collected = smth_reduced.collect()
```
(a) Implement the word count example using PySpark

```python
from pyspark import SparkConf, SparkContext

sparkConf = (SparkConf()
    .setAppName("WordCount")
    .setMaster("local"))
sc = SparkContext(conf=sparkConf)

words = sc.parallelize(["scala","java","hadoop","spark","akka"])

#assign key to each word
wordsMapped1 = words.map(lambda word: (word, 1))

#returns an RDD object
wordCount1 = wordsMapped1.reduceByKey(lambda c1, c2: c1 + c2)

print('Number of occurrences per word: ', wordCount1.collect())
```
Assignment 6-2

(a) + (b) matrix-matrix multiplication using Spark

RECAP: Assignment 4-1:

Following steps are required for performing a matrix-matrix multiplication using MapReduce:

1. Map
   
   \[(i, j, a_{ij}) \rightarrow (j, (A, i, a_{ij})) \quad (j, k, b_{jk}) \rightarrow (j, (B, k, b_{jk}))\]

2. Join
   
   \[(j, (A, i, a_{ij})) \bowtie (j, (B, k, b_{jk})) \rightarrow (j, [(A, i, a_{ij}), (B, k, b_{jk})])\]

3. Map
   
   \[(j, [(A, i, a_{ij}), (B, k, b_{jk})]) \rightarrow ((i, k), (a_{ij}b_{jk}))\]

4. ReduceByKey
   
   \[((i, k), [(a_{ij}b_{jk})]) \rightarrow ((i, k), \sum(a_{ij}, b_{jk}))\]
Assignment 6-2

Now let’s have a look at the code...

\[
\begin{bmatrix}
\cos 90^\circ & \sin 90^\circ \\
-\sin 90^\circ & \cos 90^\circ
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2
\end{bmatrix}
= \begin{bmatrix}
a_1 \\
a_2
\end{bmatrix}
\]

https://xkcd.com/184/
1. Run master node:
   * spark-class.cmd org.apache.spark.deploy.master.Master
   * spark-class.cmd is located in the bin directory

2. Check if Spark is up-and-running:
   * fire up your browser and type in: http://localhost:8080
   * you should see a Spark Master page
   * get the URL! (e.g. spark://10.153.51.36:7077)

3. Run worker node:

4. Run Python script:
   * spark-submit awesomescript.py