Pyspark week-10
================

BigData campaign data in the spark with scala

```
from pyspark import SparkContext

sc = SparkContext("local[*]","KeywordAmount")

initial_rdd = sc.textFile("/Users/trendytech/Desktop/data/bigdata-campaign-data.csv")

mapped_input = initial_rdd.map(lambda x: (float(x.split(",")[10]),x.split(",")[0]))

words = mapped_input.flatMapValues(lambda x: x.split(" "))

final_mapped = words.map(lambda x: (x[1].lower(),x[0]))

total = final_mapped.reduceByKey(lambda x,y: x+y)

sorted = total.sortBy(lambda x: x[1],False)

result = sorted.take(20)

for x in result:
   print(x)
```

=========

```
from pyspark import SparkContext

def loadBoringWords():
   boring_words = set(line.strip() for line in
open("/Users/trendytech/Desktop/data/boringwords.txt"))
   return boring_words

sc = SparkContext("local[*]","KeywordAmount")

name_set = sc.broadcast(loadBoringWords())

initial_rdd = sc.textFile("/Users/trendytech/Desktop/data/bigdata-campaign-data.csv")

mapped_input = initial_rdd.map(lambda x: (float(x.split(",")[10]),x.split(",")[0]))
```

```
words = mapped_input.flatMapValues(lambda x: x.split(" "))

final_mapped = words.map(lambda x: (x[1].lower(),x[0]))

filtered_rdd = final_mapped.filter(lambda x: x[0] not in name_set.value)

total = filtered_rdd.reduceByKey(lambda x,y: x+y)

sorted = total.sortBy(lambda x: x[1],False)

result = sorted.take(20)

for x in result:
    print(x)
```

Accumulator example
====================

```
from pyspark import SparkContext

def blankLineChecker(line):
    if(len(line) == 0):
        myaccum.add(1)


sc = SparkContext("local[*]","AccumulatorExample")

myrdd = sc.textFile("/Users/trendytech/Desktop/data/samplefile.txt")

myaccum = sc.accumulator(0.0)

myrdd.foreach(blankLineChecker)

print(myaccum.value)
```

=======

you can use foreach on a rdd but not on a local variable example list

a = rdd.collect

======

```
from pyspark import SparkContext

sc = SparkContext("local[*]", "logLevelCount")

sc.setLogLevel("INFO")

if __name__ == "__main__":
   my_list = ["WARN: Tuesday 4 September 0405",
   "ERROR: Tuesday 4 September 0408",
   "ERROR: Tuesday 4 September 0408",
   "ERROR: Tuesday 4 September 0408",
   "ERROR: Tuesday 4 September 0408",
   "ERROR: Tuesday 4 September 0408"]

   original_logs_rdd = sc.parallelize(my_list)

else:
   original_logs_rdd = sc.textFile("/Users/trendytech/Desktop/data/logsample.txt")
   print("inside the else part")

new_pair_rdd = original_logs_rdd.map(lambda x:(x.split(":")[0],1))

resultant_rdd = new_pair_rdd.reduceByKey(lambda x,y: x+y)

result = resultant_rdd.collect()

for x in result:
   print(x)
```

=========

bigLog.txt 10 million log level entries

groupByKey

reduceByKey

```
from pyspark import SparkContext

# Set the log level to only print errors
```

```
sc = SparkContext("local[*]", "LogLevelCount")

sc.setLogLevel("INFO")

# Create a SparkContext using every core of the local machine

base_rdd = sc.textFile("/Users/trendytech/Desktop/data/bigLog.txt")

mapped_rdd = base_rdd.map(lambda x: (x.split(":")[0], x.split(":")[1]))

grouped_rdd =  mapped_rdd.groupByKey()

final_rdd = grouped_rdd.map(lambda x: (x[0], len(x[1])))

result = final_rdd.collect()

for x in result:
   print(x)
```

============

```
from pyspark import SparkContext

sc = SparkContext("local[*]", "LogLevelCount")

sc.setLogLevel("INFO")

base_rdd = sc.textFile("/Users/trendytech/Desktop/data/bigLog.txt")

mapped_rdd = base_rdd.map(lambda x: (x.split(":")[0], 1))

reduced_rdd =  mapped_rdd.reduceByKey(lambda x,y: x+y)

result = reduced_rdd.collect()

for x in result:
   print(x)
```

==========

Miscellaneous things
=====================

1)

scala
=======
```
val a = 1 to 100
val base = sc.parallelize(a)
base.reduce((x,y) => x+y)
```

pyspark
=========
```
a = range(1,101)
base = sc. parallelize(a)
base.reduce(lambda x,y: x+y)
```

2)

```
input = sc.textFile("/Users/trendytech/Desktop/data/customer-orders.csv")

input.saveAsTextFile("/Users/trendytech/Desktop/data/output10")
```

3. Count - this is an action and works the same way as we saw in scala codes.

4. sc.defaultParallelism

5. get the num of partitions in an rdd

```
rdd.getNumPartitions()
```

6.my_list = ("WARN: Tuesday 4 September 0405",
"ERROR: Tuesday 4 September 0408",
"ERROR: Tuesday 4 September 0408",
"ERROR: Tuesday 4 September 0408",
"ERROR: Tuesday 4 September 0408",
"ERROR: Tuesday 4 September 0408")

```
original_logs_rdd = sc.parallelize(my_list)
original_logs_rdd.getNumPartitions()
```

7) sc.defaultMinPartitions - 2

8) repartition

9) coalesce