



# Assignment

**Week15:** Apache Spark - Streaming  
Part-1

# IMPORTANT

## Self-assessment enables students to develop:

1. A sense of responsibility for their own learning and the ability & desire to continue learning,
2. Self-knowledge & capacity to assess their own performance critically & accurately, and
3. An understanding of how to apply their knowledge and abilities in different contexts.

All assignments are for self-assessment. Solutions will be released on every subsequent week. Once the solution is out, evaluate yourself.

No discussions/queries allowed on assignment questions in slack channel.

**Note:** You can raise your doubts in the subsequent week once the solution is released

TRENDYTECH 9108179578

## Solution1:

```
import org.apache.log4j.Level
import org.apache.log4j.Logger
import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext

object W15_Problem1 extends App {

  Logger.getLogger("org").setLevel(Level.ERROR)

  val sc = new SparkContext("local[*]", "SearchDataWithReduceByKeyAndWindow")

  //creating spark Streaming Context
  val ssc = new StreamingContext(sc, Seconds(2))

  //lines is DStream
  val lines = ssc.socketTextStream("localhost", 1724)

  ssc.checkpoint(".")
```

TRENDYTECH 9108179578

```
/**THESE "NAMED FUNCTION" FOR "reduceByKeyAndWindow" */
def summaryFunc(x: Int, y: Int) = { x + y }

def inverseFunc(x: Int, y: Int) = { x - y }

//words is a transformed DStream
val words = lines.flatMap(x => x.split(" ")).map(x => x.toLowerCase())

val pairs = words.map(x => (x, 1)).filter(a => a._1.startsWith("big"))

/**these "reduceByKeyAndWindow" with Named Function is a STATEFUL TRANSFORMATION & is working
on FEW RDD's*/
val wordCounts = pairs.reduceByKeyAndWindow(summaryFunc(_, _), inverseFunc(_, _), Seconds(10),
Seconds(4))

wordCounts.print()

ssc.start()

ssc.awaitTermination()

}
```



TRENDYTECH 9108179578

## Solution 2:

```
import org.apache.log4j.Level
import org.apache.log4j.Logger
import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
```

```
object W15_Problem_2 extends App {
```

```
  Logger.getLogger("org").setLevel(Level.ERROR)
```

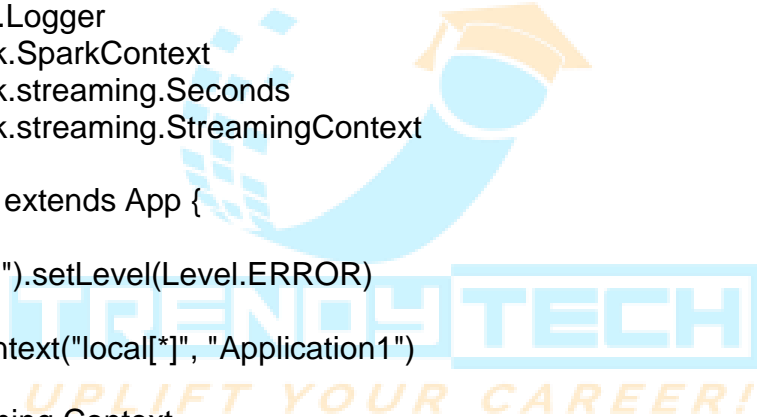
```
  val sc = new SparkContext("local[*]", "Application1")
```

```
  //creating spark Streaming Context
```

```
  val ssc = new StreamingContext(sc, Seconds(2))
```

```
  //lines is DStream
```

```
  val lines = ssc.socketTextStream("localhost", 9544)
```



TRENDYTECH 9108179578

```
ssc.checkpoint(".")

/**THESE "NAMED FUNCTION" FOR "reduceByWindow" */

def summaryFunc(x: String, y: String) = { (x.toInt + y.toInt).toString() }

def inverseFunc(x: String, y: String) = { (x.toInt - y.toInt).toString() }

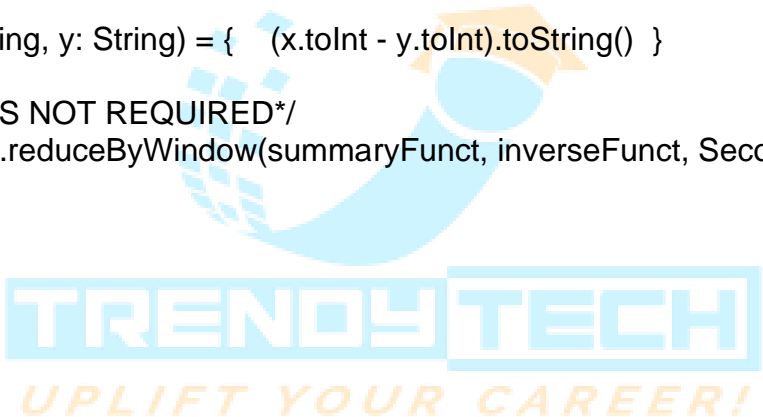
/**here PAIRED RDD IS NOT REQUIRED*/
val wordCounts = lines.reduceByWindow(summaryFunc, inverseFunc, Seconds(10), Seconds(2))

wordCounts.print()

ssc.start()

ssc.awaitTermination()

}
```



TRENDYTECH 9108179578

## Solution 3:

```
import org.apache.log4j.Level
import org.apache.log4j.Logger
import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
```

```
object W15_Problem_3 extends App {
```

```
  Logger.getLogger("org").setLevel(Level.ERROR)
```

```
  val sc = new SparkContext("local[*]", "Application1")
```

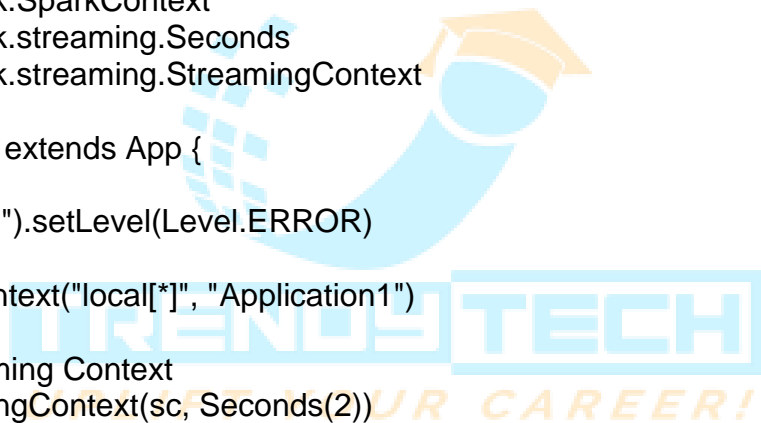
```
  //creating spark Streaming Context
```

```
  val ssc = new StreamingContext(sc, Seconds(2))
```

```
  //lines is DStream
```

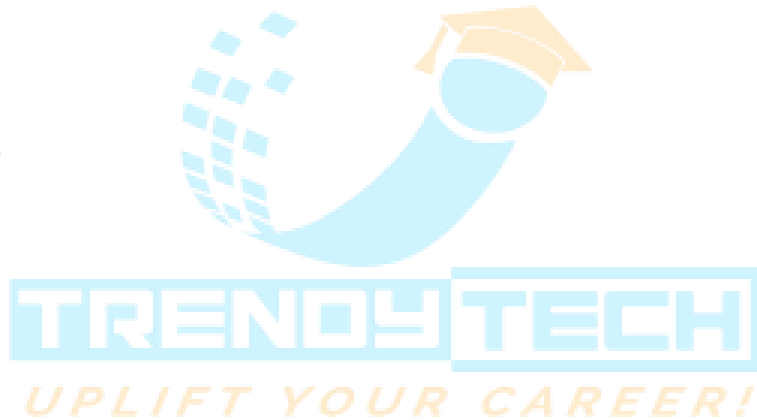
```
  val lines = ssc.socketTextStream("localhost", 9544)
```

```
  ssc.checkpoint(".")
```



TRENDYTECH 9108179578

```
/**IT'LL COUNT the NO. of Lines in the window */  
val lineCounts = lines.countByWindow(Seconds(10), Seconds(2))  
  
lineCounts.print()  
  
ssc.start()  
  
ssc.awaitTermination()  
  
}
```



TRENDYTECH 9108179578





**5** Star Google Rated  
Big Data Course

**LEARN FROM THE EXPERT**



**9108179578**

**Call for more details**