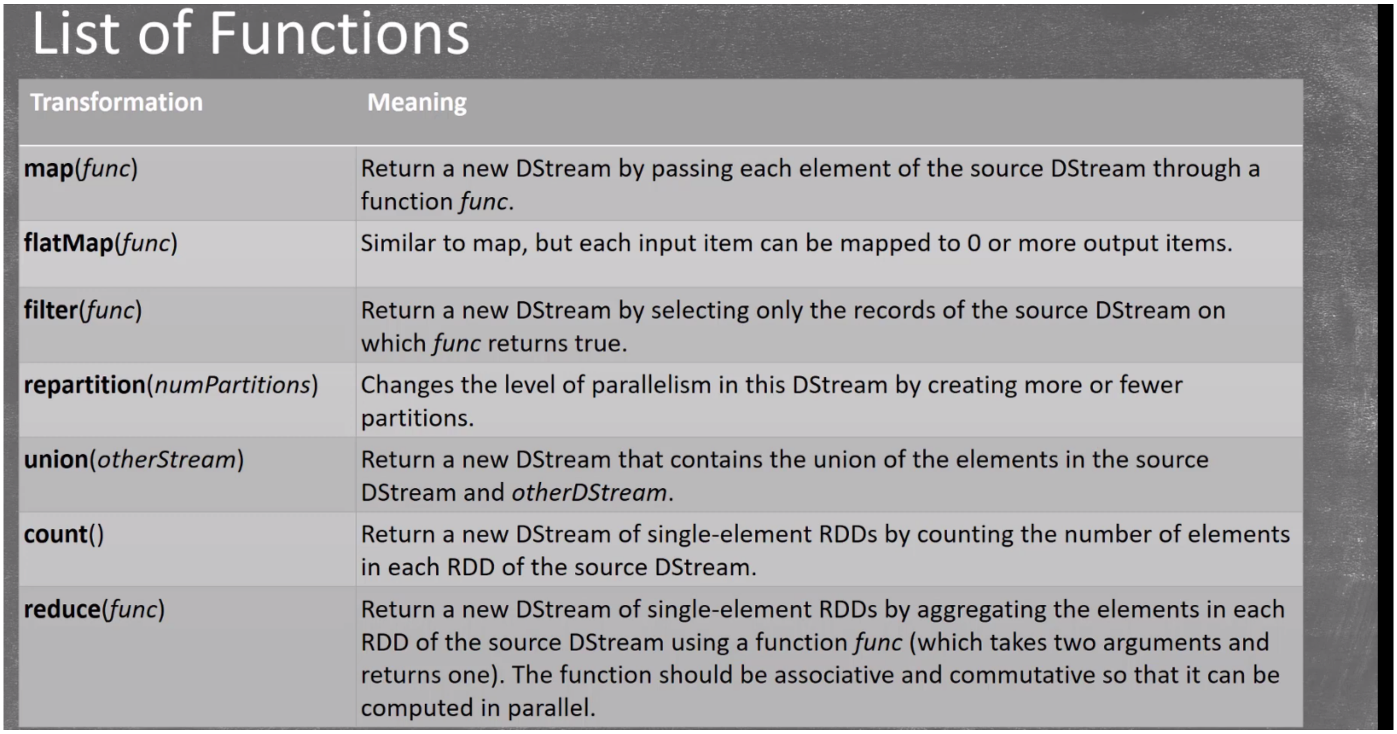
In [19]:

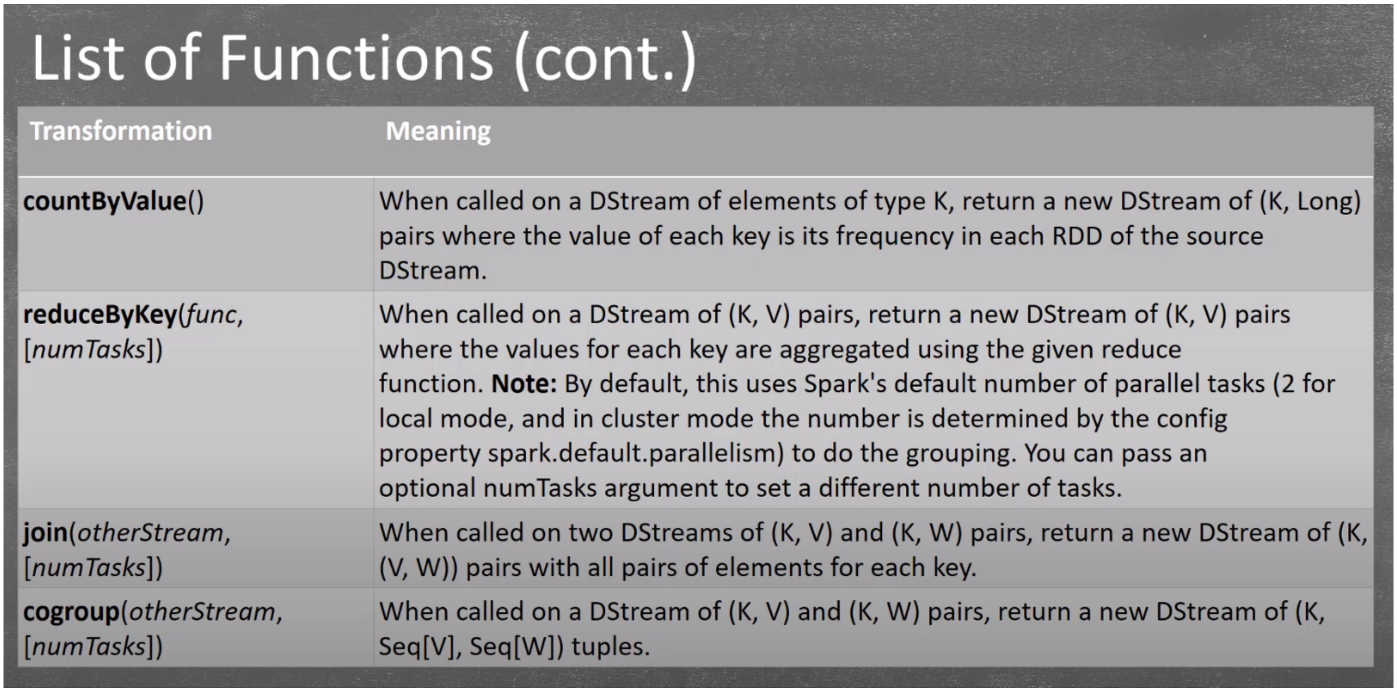
```
import findspark
from pyspark.streaming import StreamingContext
from pyspark.sql import SparkSession
import math

findspark.init()
```



List of Functions

Transformation	Meaning
map(*func*)	Return a new DStream by passing each element of the source DStream through a function *func*.
flatMap(*func*)	Similar to map, but each input item can be mapped to 0 or more output items.
filter(*func*)	Return a new DStream by selecting only the records of the source DStream on which *func* returns true.
repartition(*numPartitions*)	Changes the level of parallelism in this DStream by creating more or fewer partitions.
union(*otherStream*)	Return a new DStream that contains the union of the elements in the source DStream and *otherDStream*.
count()	Return a new DStream of single-element RDDs by counting the number of elements in each RDD of the source DStream.
reduce(*func*)	Return a new DStream of single-element RDDs by aggregating the elements in each RDD of the source DStream using a function *func* (which takes two arguments and returns one). The function should be associative and commutative so that it can be computed in parallel.



List of Functions (cont.)

Transformation	Meaning
countByValue()	When called on a DStream of elements of type K, return a new DStream of (K, Long) pairs where the value of each key is its frequency in each RDD of the source DStream.
reduceByKey(*func*, [*numTasks*])	When called on a DStream of (K, V) pairs, return a new DStream of (K, V) pairs where the values for each key are aggregated using the given reduce function. **Note:** By default, this uses Spark's default number of parallel tasks (2 for local mode, and in cluster mode the number is determined by the config property spark.default.parallelism) to do the grouping. You can pass an optional numTasks argument to set a different number of tasks.
join(*otherStream*, [*numTasks*])	When called on two DStreams of (K, V) and (K, W) pairs, return a new DStream of (K, (V, W)) pairs with all pairs of elements for each key.
cogroup(*otherStream*, [*numTasks*])	When called on a DStream of (K, V) and (K, W) pairs, return a new DStream of (K, Seq[V], Seq[W]) tuples.

In [20]:

```
try: ssc.stop(True, True)
except: pass
try: spark.stop()
except: pass
```

In [21]:

```
spark=SparkSession.builder.appName("SparkStreaming-03").master('local[1]').getOrCreate()
sc=spark.sparkContext
# .config("spark.driver.allowMultipleContexts","true")
spark
```

Out[21]: **SparkSession - in-memory**

SparkContext

Spark UI

```
Version     v2.4.8
Master      local[1]
AppName     SparkStreaming-03
```

In [10]:

```
ssc=StreamingContext(sc, 1)
ssc # 1=1 second
```

Out[10]:

```
<pyspark.streaming.context.StreamingContext at 0x229e0bbd788>
```

In [11]:

```
lines=ssc.socketTextStream("localhost", 8000)
lines.pprint()
```

In [12]:

```
ssc.start()
```

```
-------------------------------------------
Time: 2021-09-18 12:32:40
-------------------------------------------
```

In [ ]: