Algorithms and Applications of Data Mining

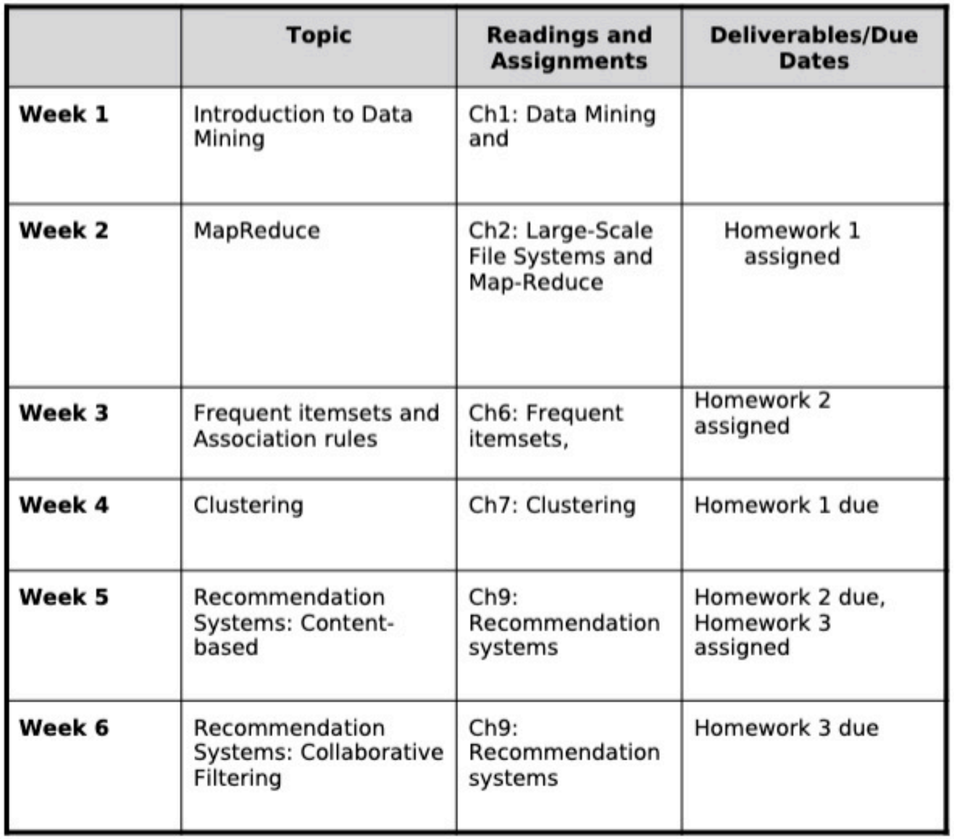
Yijun Lin

yijunlin@usc.edu

01/31

About This Course

- Spring 2021, Friday, 6-8 PM PST
- Instructor: Yao-Yi Chiang
- TA: Yijun Lin
  - Office Hour Sat. 7-9 PM PST
- Syllabus:



	Topic	Readings and Assignments	Deliverables/Due Dates
Week 1	Introduction to Data Mining	Ch1: Data Mining and	
Week 2	MapReduce	Ch2: Large-Scale File Systems and Map-Reduce	Homework 1 assigned
Week 3	Frequent itemsets and Association rules	Ch6: Frequent itemsets,	Homework 2 assigned
Week 4	Clustering	Ch7: Clustering	Homework 1 due
Week 5	Recommendation Systems: Content-based	Ch9: Recommendation systems	Homework 2 due, Homework 3 assigned
Week 6	Recommendation Systems: Collaborative Filtering	Ch9: Recommendation systems	Homework 3 due

Assignments

- Theoretical and programming questions
  - Real-world datasets

- Homework 1 - basic spark operations
- Homework 2 - mining frequent itemset
- Homework 3 - recommender system

- Optional – clustering

Config Environment

- **Python** is required for all the assignments

- Implementing with Apache Spark Framework
  - python=3.7
  - pyspark=3.0.1
  - git clone https://github.com/linyijun/cis-data-mining-ta-materials

- Install miniconda/anaconda
  - conda env create -f spark-env.yml python=3.7

- Install PyCharm

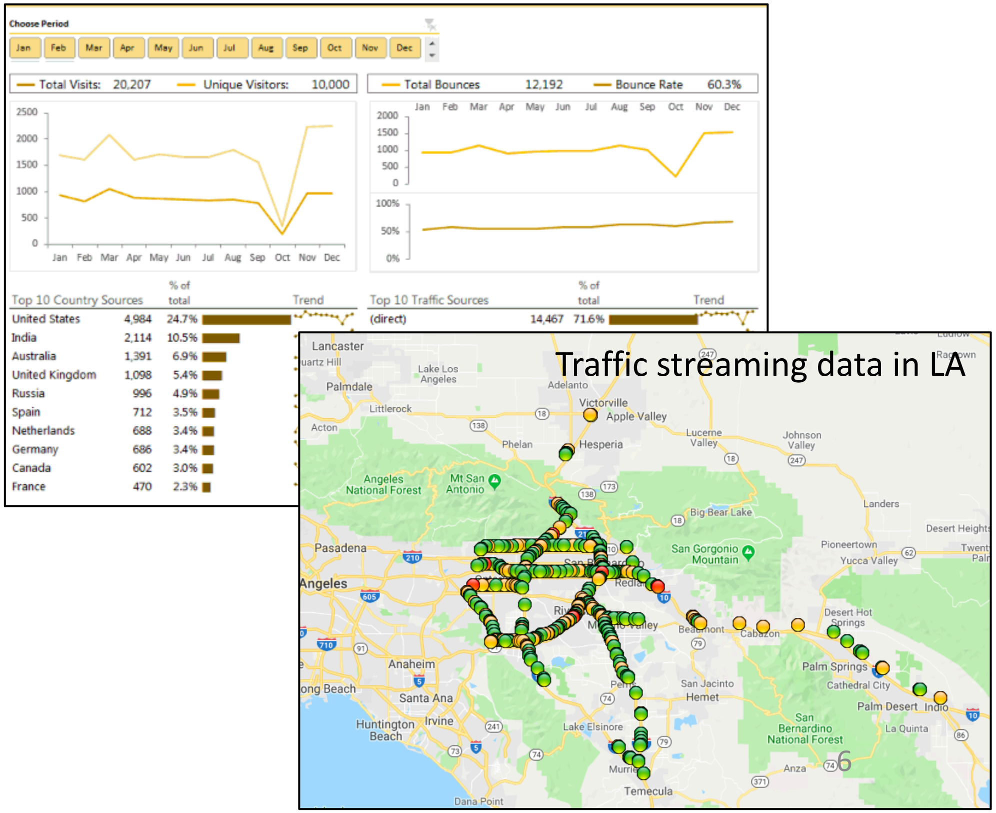
Introduction to Spark

Thanks for source slide and material to: Dr. Heather Miller
https://www.coursera.org/learn/scala-spark-big-data/home/welcome

What is Spark?

- **Apache Spark** is a unified analytics engine for large-scale data processing

- Application areas
  - Interactive Data Query
  - Real-time Data Analysis
  - Streaming Data Processing



Choose Period

Total Visits: 20,207 Unique Visitors: 10,000

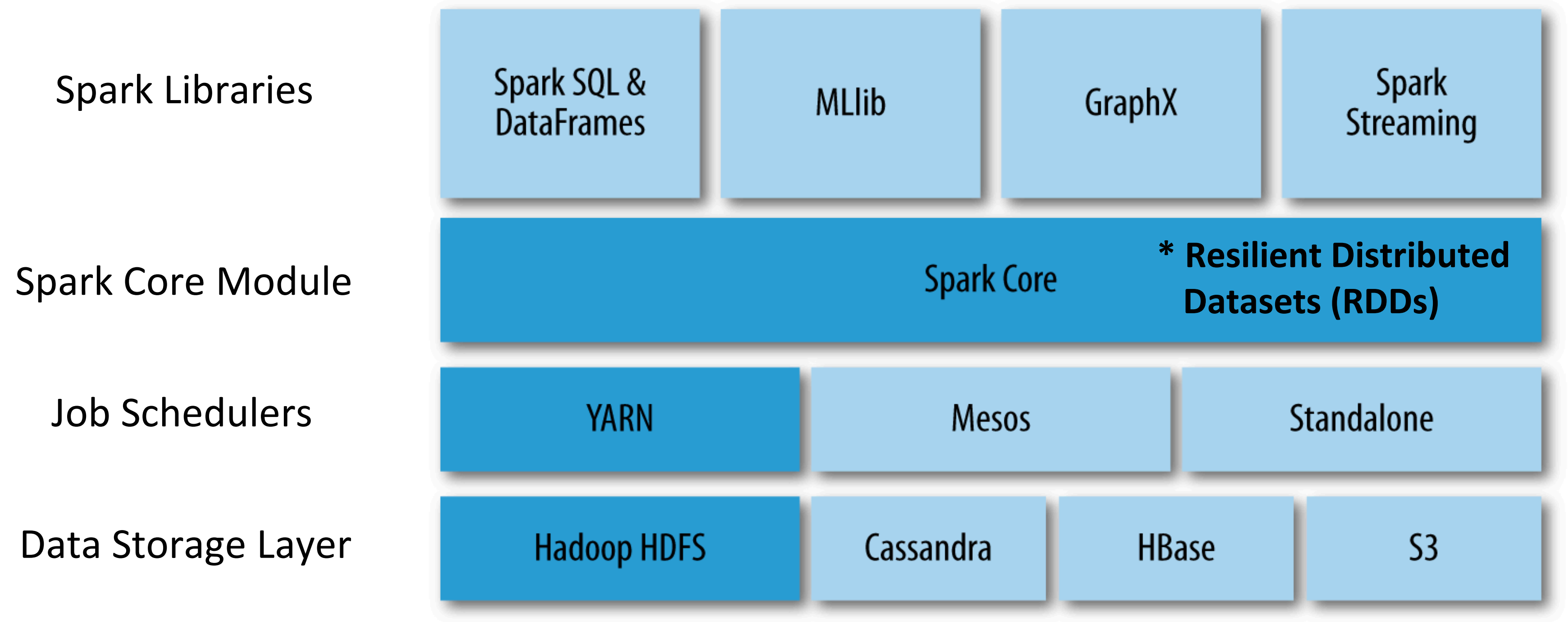
Total Bounces 12,192 Bounce Rate 60.3%

Top 10 Country Sources		% of total	Trend
United States	4,984	24.7%	
India	2,114	10.5%	
Australia	1,391	6.9%	
United Kingdom	1,098	5.4%	
Russia	996	4.9%	
Spain	712	3.5%	
Netherlands	688	3.4%	
Germany	686	3.4%	
Canada	602	3.0%	
France	470	2.3%	

Top 10 Traffic Sources		% of total	Trend
(direct)	14,467	71.6%	

Traffic streaming data in LA

Spark Stack

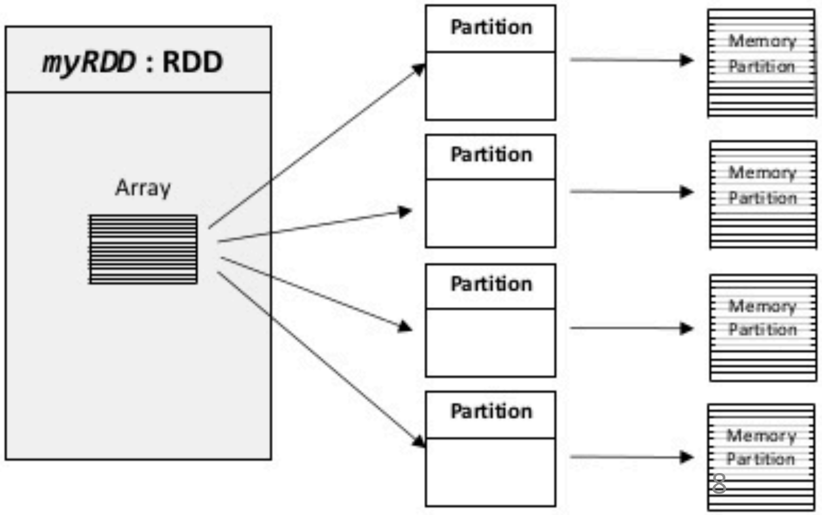


Spark Libraries
Spark SQL & DataFrames
MLlib
GraphX
Spark Streaming
Spark Core Module
Spark Core
* Resilient Distributed Datasets (RDDs)
Job Schedulers
YARN
Mesos
Standalone
Data Storage Layer
Hadoop HDFS
Cassandra
HBase
S3

Resilient Distributed Datasets (RDDs)

- An RDD is an **immutable**, **in-memory collection** of objects
- Each RDD can be split into multiple partitions, which in turn are computed on different nodes of the cluster

- RDDs seem a lot like Scala collections
  - RDD[T] and List[T]

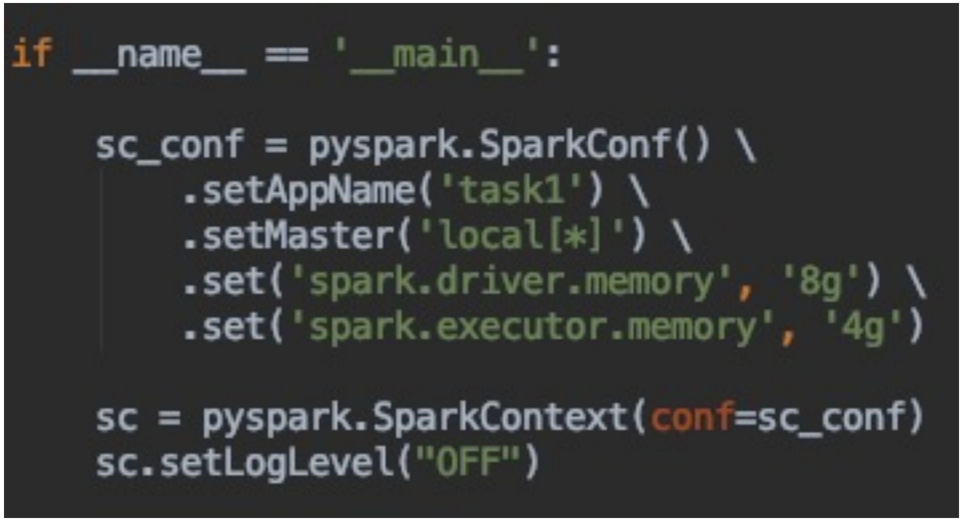


How to create an RDD

- RDDs can be created in two ways:
  - Creating from a SparkContext object
  - Transforming from an existing RDD

How to create an RDD (Cont.)

- Creating from a **SparkContext** object
  - Can be thought as your handle to the Spark cluster
  - Represents the connection to a Spark cluster



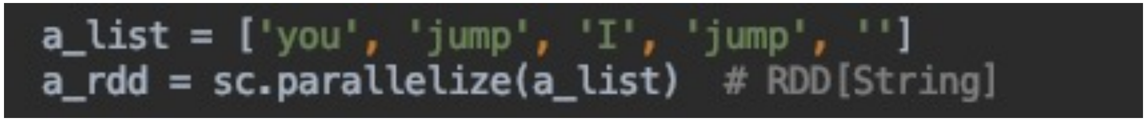
```
if __name__ == '__main__':

    sc_conf = pyspark.SparkConf() \
        .setAppName('task1') \
        .setMaster('local[*]') \
        .set('spark.driver.memory', '8g') \
        .set('spark.executor.memory', '4g')

    sc = pyspark.SparkContext(conf=sc_conf)
    sc.setLogLevel("OFF")
```

How to create an RDD (Cont.)

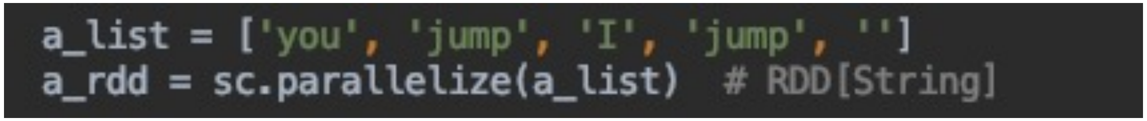
- Creating from a **SparkContext** object
  - **parallelize:** convert a local Scala collection to an RDD



```
a_list = ['you', 'jump', 'I', 'jump', '']
a_rdd = sc.parallelize(a_list)  # RDD[String]
```

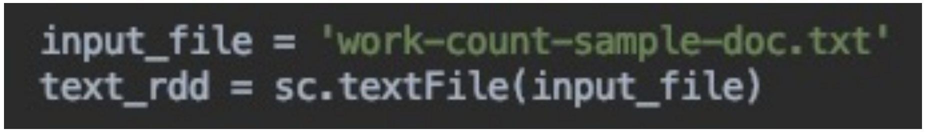
How to create an RDD (Cont.)

- Creating from a **SparkContext** object
  - **parallelize:** convert a local Scala collection to an RDD



```
a_list = ['you', 'jump', 'I', 'jump', '']
a_rdd = sc.parallelize(a_list)  # RDD[String]
```

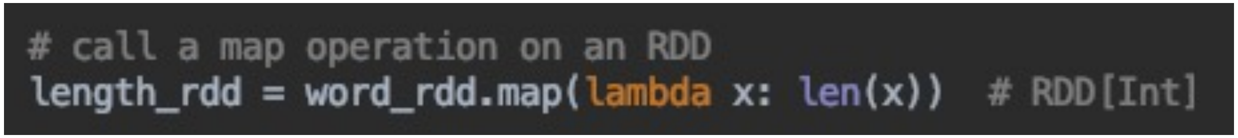
  - **textFile**: read a file from HDFS or local file system



```
input_file = 'work-count-sample-doc.txt'
text_rdd = sc.textFile(input_file)
```

How to create an RDD (Cont.)

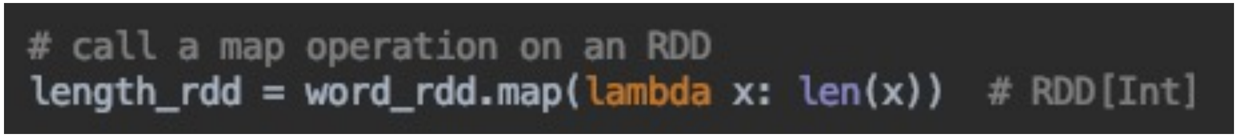
- Transforming from an existing RDD
  - E.g., calling a *map operation* on an existing RDD, it will return a new RDD



```
# call a map operation on an RDD
length_rdd = word_rdd.map(lambda x: len(x))  # RDD[Int]
```

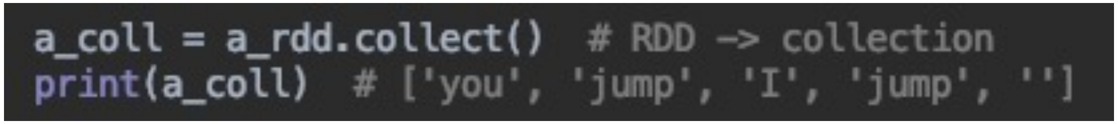
RDD Operations

- Transformations
  - E.g., map, filter, ...



```
# call a map operation on an RDD
length_rdd = word_rdd.map(lambda x: len(x))  # RDD[Int]
```

- Actions
  - E.g., collect, reduce ...



```
a_coll = a_rdd.collect()  # RDD -> collection
print(a_coll)  # ['you', 'jump', 'I', 'jump', '']
```

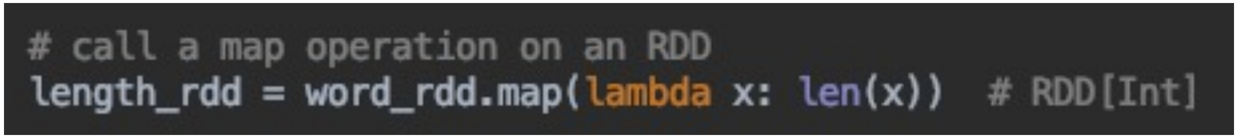
Transformations VS Actions

- Transformations
  - Return new RDDs as results
  - They are **lazy**, the result RDD is not immediately computed

- Actions
  - Compute a result based on an RDD, and returned
  - They are **eager**, the result is immediately computed

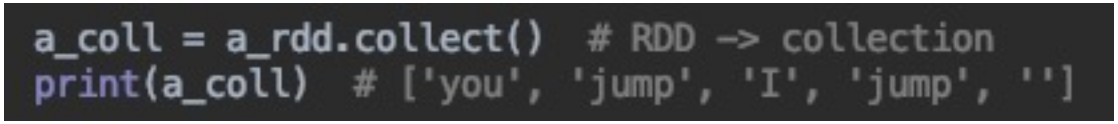
Transformations VS Actions

- Transformations
  - Return new RDDs as results
  - They are **lazy**, the result RDD is not immediately computed



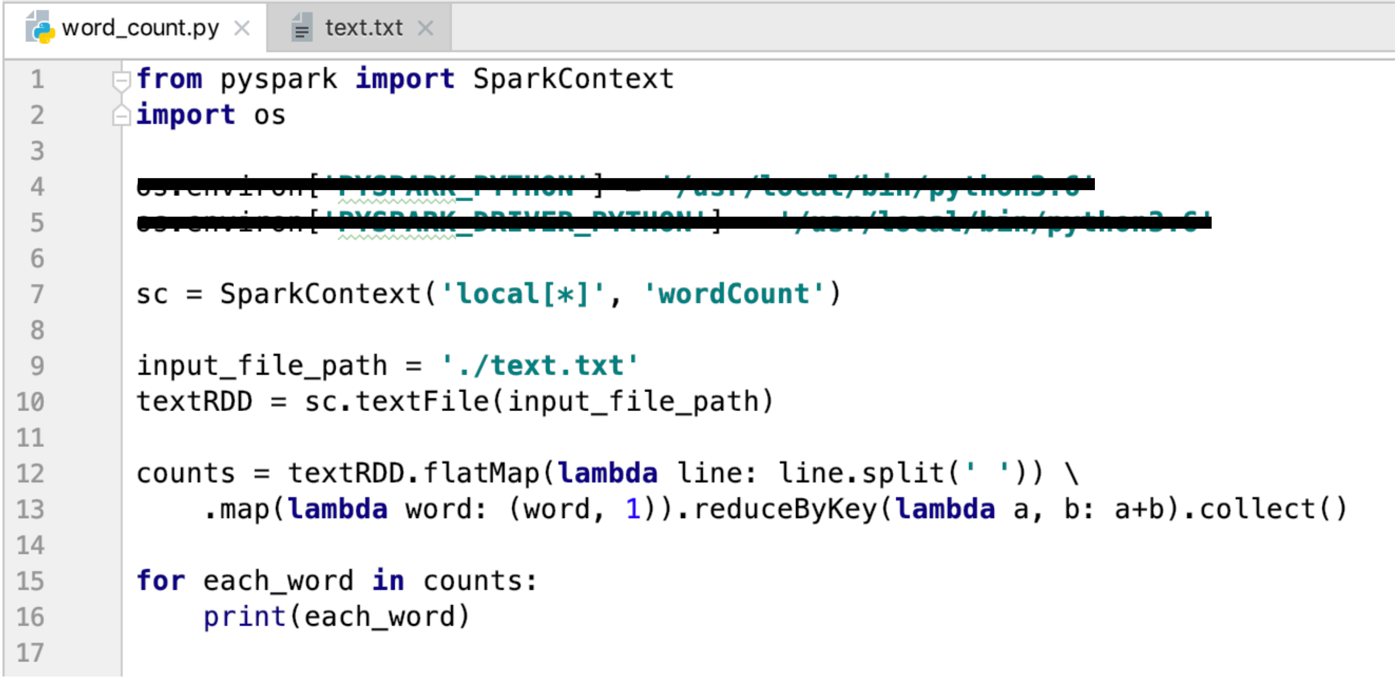
```
# call a map operation on an RDD
length_rdd = word_rdd.map(lambda x: len(x))  # RDD[Int]
```

- Actions
  - Compute a result based on an RDD, and returned
  - They are **eager**, the result is immediately computed



```
a_coll = a_rdd.collect()  # RDD -> collection
print(a_coll)  # ['you', 'jump', 'I', 'jump', '']
```

Word Count



```
from pyspark import SparkContext
import os

[redacted]

sc = SparkContext('local[*]', 'wordCount')

input_file_path = './text.txt'
textRDD = sc.textFile(input_file_path)

counts = textRDD.flatMap(lambda line: line.split(' ')) \
    .map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b).collect()

for each_word in counts:
    print(each_word)

```

If you want to learn more…

- Official documentation
  - http://spark.apache.org/docs/latest/

- Online course
  - Coursera: Big Data Analysis with Scala and Spark

- Books
  - *Learning Spark, O' Reilly*
  - Advanced Analytics with Spark: Patterns for Learning from Data at Scale, *O' Reilly*
  - *Machine Learning with Spark, Packt*