**A TUTORIAL ON SPARK**

1. **CodeBig: Apache Spark Hands On Labs**

Sejal Chauhan, Lyujia (Chloe) Zhang, Shubhi Rani, Wan Chen

1. **Agenda:**

This Hands On Labs will help you install and write applications using Apache Spark. You will get an idea on how to use Spark for SQL, Machine Learning, Graph Processing and Stream Analytics step-by-step: Three level- Beginner, Intermediate, Expert with example code and corresponding exercise giving intermittent hands-on experience.

* Install and setup environment for Spark
* Spark for SQL
* Spark for ML
* Spark for Stream Analytics

1. **Logistics and Prerequisites:**

* Pair two students so that they can hack and collaborate together
* TA assignment:
  + TA notes available
* 64-bit Linux (Ubuntu, Debian, CentOS, RHEL)
* VMware Fusion / Workstation to run Linux
* Oracle JDK
* Knowledge of Scala/ Java or Python.

1. **Abstract:**

People with some experience with SQL queries or machine learning workloads can go forth and get their parallel clusters running on spark with this tutorial.

This gives a quick overview of Spark and its related technologies along with getting hands-on knowledge.

1. **Takeaways:**
2. Get to know about Spark local setup
3. Running Spark SQL
4. Working with MLib on Spark
5. Hands-on exercises with Debugging tips
6. **Contents:**

Section 1: Introduction

Architecture overview

Section 2: Environment setup and Installation

Running examples

Section 3: SQL & Spark Streaming

Example - SQL on dataframe

Exercise - SQL on parquet

Spark Streaming example project

Section 4 : Machine Learning

MLlib

Tic Tac Linear Regression

1. **Architecture overview**

* **Introduction to Spark RDDs:**

**RDD: Resilient Distributed Datasets**

Its simply a representation of a dataset which is distributed across a cluster.

Distributed datasets: create a dataset from data and apply parallel operations to it

**MapReduce** is a processing technique that consists of two important tasks: Map and Reduce.

**Map** - takes a data and converts it into another set of data converting the elements to tuples.

**Reduce** - takes the output obtained from Map and combines the tuples into smaller set of tuples.

**RDD API operations:**

**Transformations** : create new dataset from an already existing one. Examples are

map, filter, union, intersection and so on..

**Actions:** These return a value to the actual program after the computation is done.

Examples include reduce, count, collect and so on..

**Shared variables:**

Broadcast variables – cache a value in memory on all nodes  
 Accumulators – variables that are incremented (counters/sums)

* **Schedulers**

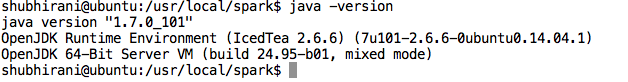
**DAGScheduler**

DAGScheduler is the scheduling layer of Apache Spark that implements stage-oriented scheduling. Basically, it does three tasks. First, it computes an execution DAG for a job. Next, it determines the preferred locations to run each task on. Last,it handles failure due to shuffle output files being lost. DAGScheduler runs stages on topological order.

**Task Scheduler**

Task Scheduler schedules tasks for a single Spark application according to scheduling mode (aka order task policy). A TaskScheduler get sets of tasks submitted to it from DAGScheduler for each stage, and is responsible for sending the tasks to the cluster, running them, retrying if there are failures, and mitigating stragglers.

1. **Install and Setup environment for Spark**

* Prerequisite: Mac laptop or Ubuntu on a virtual machine (Windows)
* Install JAVA
  + Install Open JDK
    - $sudo apt-get install openjdk-7-jdk
  + Install Oracle JAVA package
    - $sudo add-apt-repository ppa:webupd8team/java
    - $sudo apt-get update sudo apt-get install
    - $oracle-java7-installer
  + Specify the JAVA environment
    - Open /etc/environment  
      $sudo nano /etc/environment
    - Add the following line  
      JAVA\_HOME=”/usr/lib/jvm/java-7-oracle”
    - Ctrl-x to exit and type y to save the file and then hit enter.
    - Force OS to reload the /etc/environment file  
      source /etc/environment
  + Verify the JAVA environment
    - $java -version  
      
* Install SSH remote access
  + Install ssh server on worker nodes so that master can have access to these nodes
    - $sudo apt-get install openssh-server
  + Generate a master key on master node for remote access
    - $ssh-keygen
  + For worker nodes
    - $ssh copy-id -i ~/.ssh/id\_rsa.pub <uname\_on\_remote\_access>@<ip addr>
    - Ex: $*ssh copy-id -i ~/.ssh/id\_rsa.pub shubhirani@172.20.10.8*
    - Note: To get IP addr, type /sbin/ifconfig on terminal and use inet addr showing for eth0
* Download Spark
  + Download the latest spark
    - <https://spark.apache.org/downloads.html>
    - 
  + Create a folder in /usr/local
    - $mkdir /usr/local/spark
  + Extract the Spark folder
    - $tar xvf spark-2.0.0-bin-hadoop2.7.tgz -C /usr/local/spark
* Install Spark
  + Create a folder called SparkStandalone which holds the record of each worker job
  + Configure Spark Build
    - Goto /usr/local/spark folder
    - Open the ./conf/spark-env.sh file
    - Add following line to the file

export SPARK\_WORKER\_MEMORY=1g

export SPARK\_EXECUTOR\_MEMORY=512m

export SPARK\_WORKER\_INSTANCES=2

export SPARK\_WORKER\_CORES=1

export SPARK\_MASTER\_IP=<insert your ip address here>

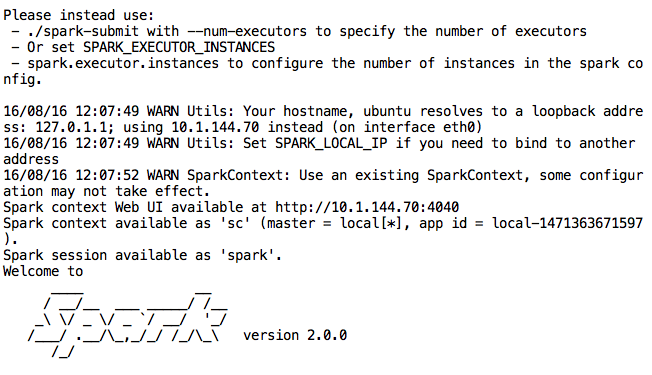
export SPARK\_WORKER\_DIR=<path of SparkStandalone folder>

* + - Rename the ./conf/slaves.template file to ./conf/slaves.sh  
      $mv ./conf/slaves.template ./conf/slaves.sh
    - Open ./conf/slaves.sh file
    - Remove the “localhost” line and add the IP address of worker nodes
      * If master IP address is 172.20.10.8, then worker address should be 173.20.10.9 (add +1)
  + Setup environment for spark
    - Open ~/.bashrc file
    - Add the following lines and save

export PATH=$PATH:/usr/local/spark/bin

* + - Source the ~/.bashrc file

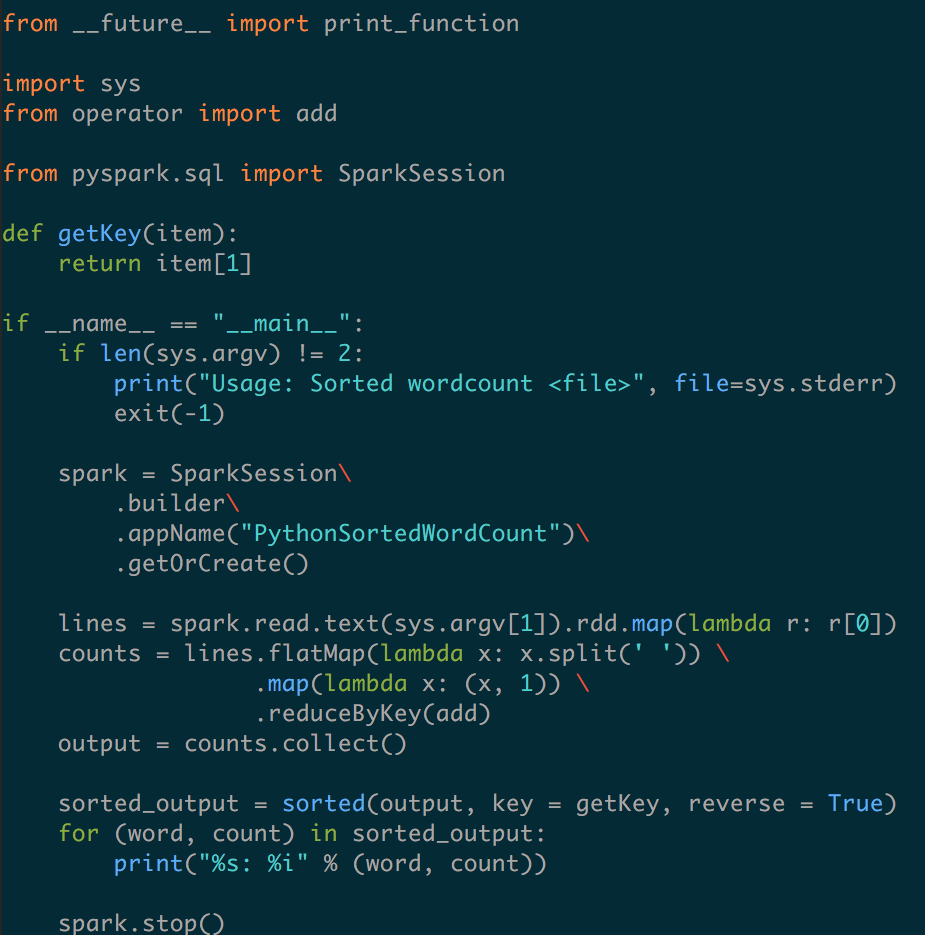
$source ~/.bashrc

* + Check the spark installation
    - $spark-shell  
      
* Running Example

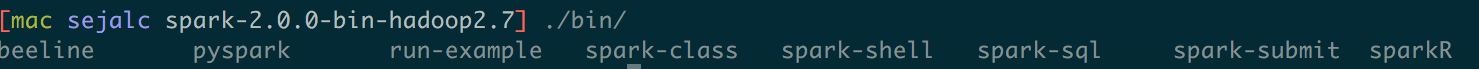
./bin/run-example org.apache.spark.examples.**SparkPi**

**Sample Code:**

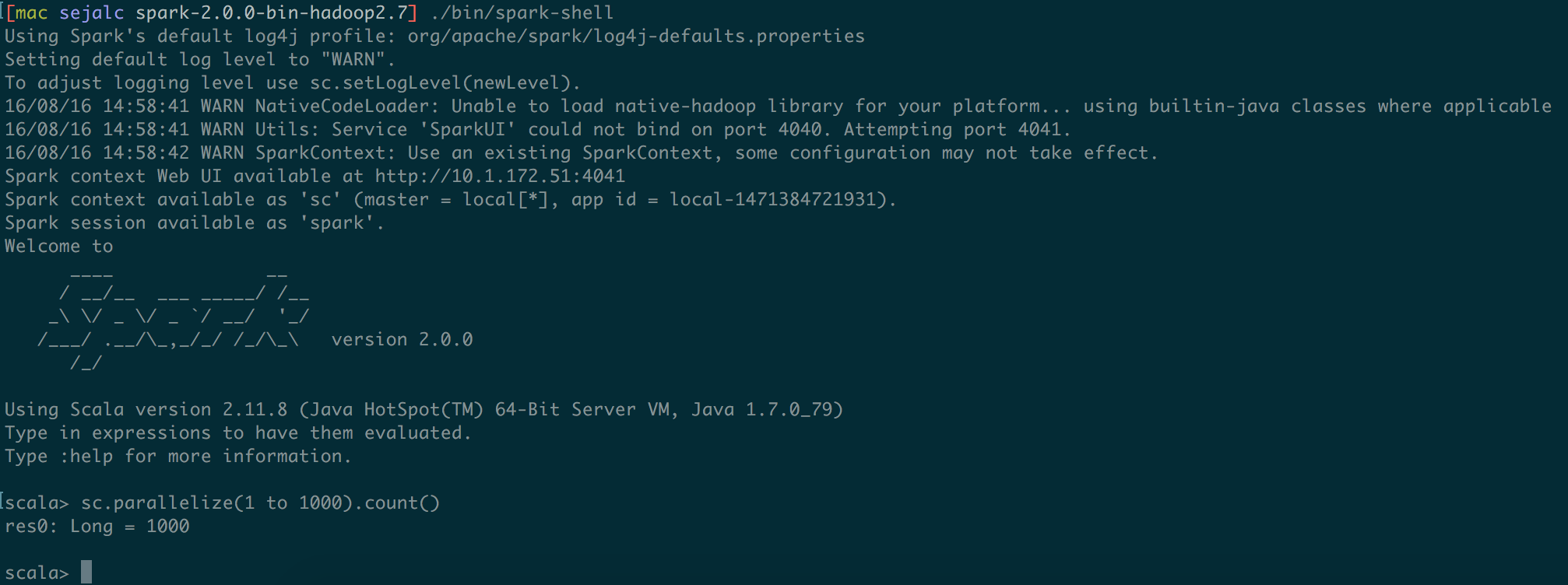
Input a file document and print the word counts in the order of decreasing frequency.



There are multiple binaries that can be utilized for various applications like Spark SQL, SparkR.

Pyspark and spark-shell will give you interactive Spark programming mode to run python and scala code.  
 

**Scala Interactive Programming:**

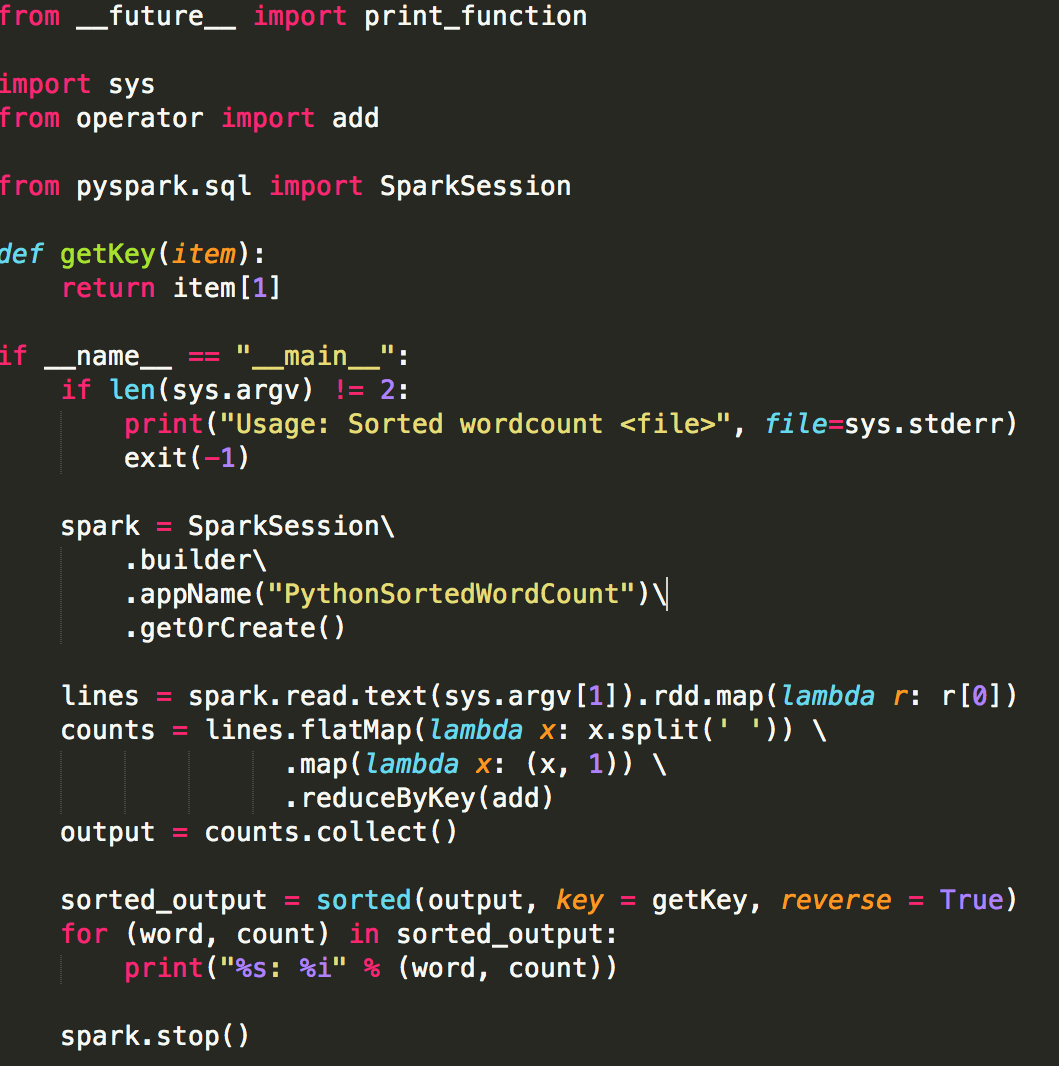


For submitting files to the spark framework spark-submit is used, like shown below.

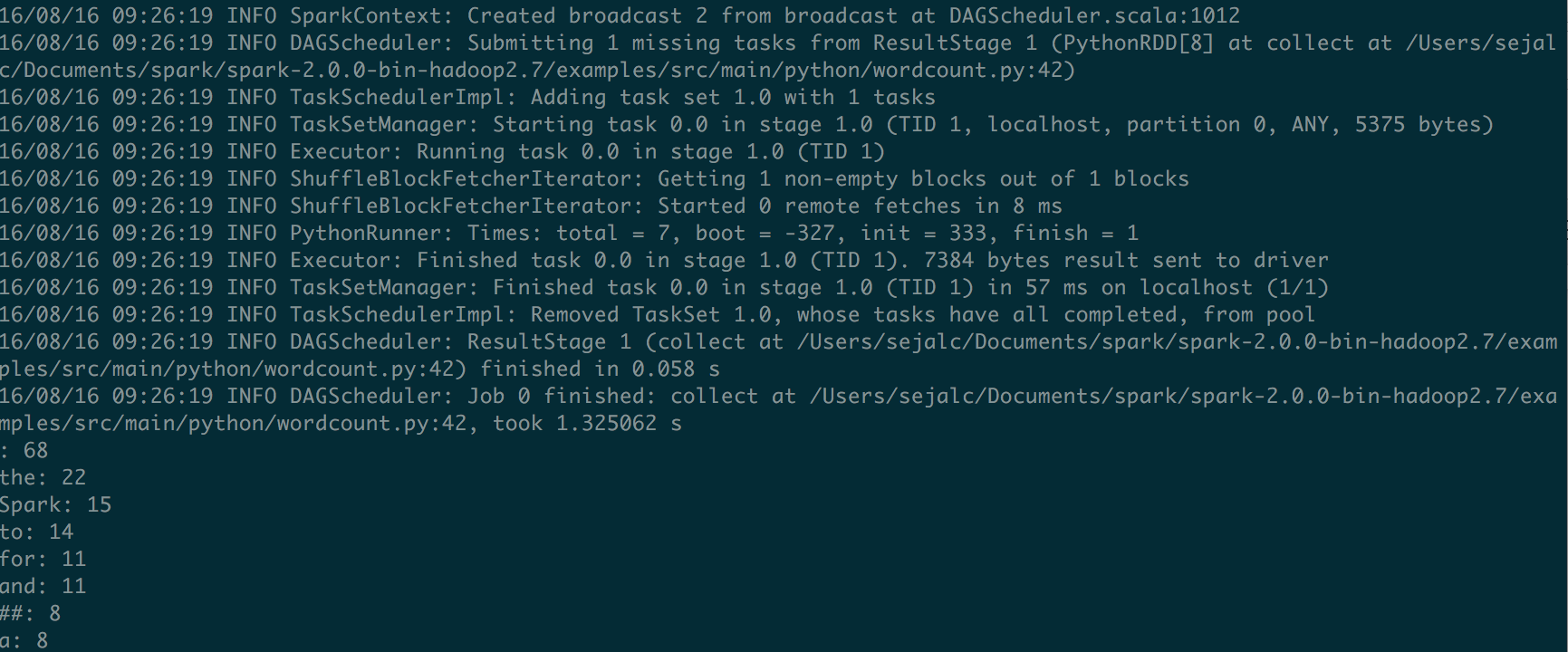
Run the source file(github link) using the following **command** in spark-2.0.0-bin-hadoop2.7 folder:

**./bin/spark-submit sortedwordcount.py <file>**

**Sample Code of sorted word count:**



**Sample output generated (Python example):**



1. **Spark SQL:**

* Spark SQL is a Spark module for structured data processing.
* There are several ways to interact with Spark SQL including SQL and the Dataset API.
* When computing a result the same execution engine is used, independent of which API/language you are using to express the computation. This unification means that developers can easily switch back and forth between different APIs based on which provides the most natural way to express a given transformation.
* **Spark SQL, Dataframes and Dataset Guide:** [**http://spark.apache.org/docs/latest/sql-programming-guide.html**](http://spark.apache.org/docs/latest/sql-programming-guide.html)

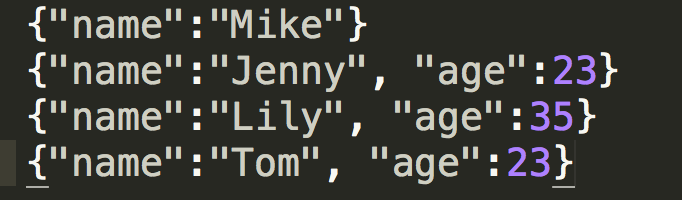
**Example:**

Given a dataset of employees list with their names and ages, do queries on it with Spark SQL as following:

1. Print out the dataframe generated by reading the json file
2. Print out attribute types by printing schema
3. Show all names listed in the dataset
4. Count number of employees at each age
5. Show all names of employees with age younger than 30

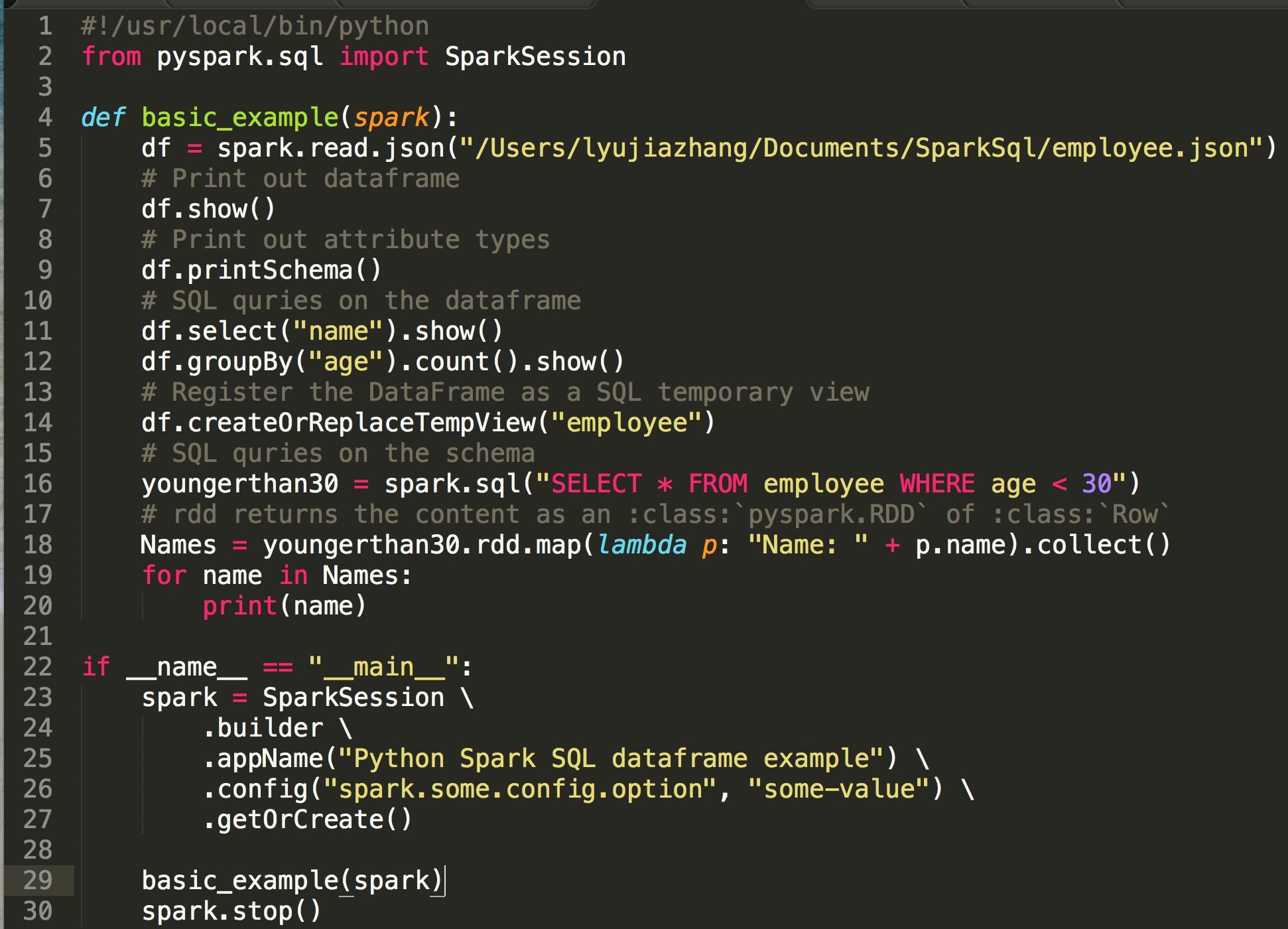
**Employee.json**

Dataset in json format, with two attributes (name, age).



**Basic\_dataframe.py**

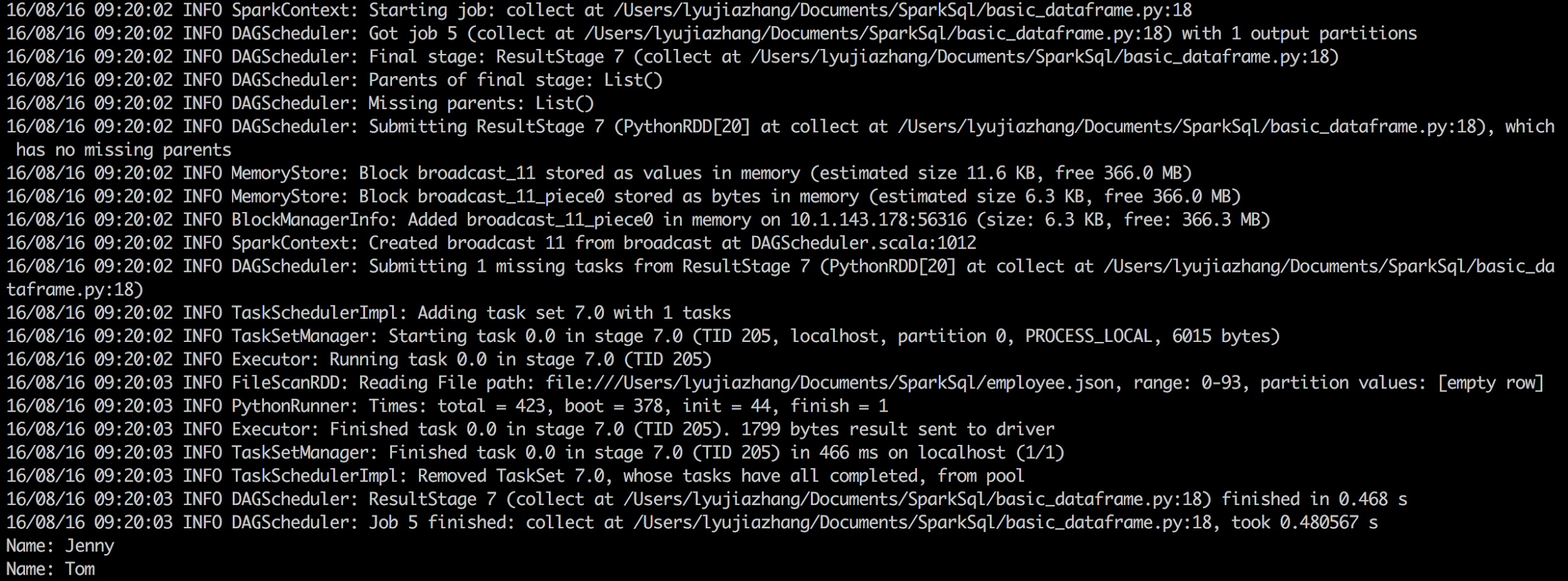
In the main function, we need to initialize a Spark session and set configuration if needed before running the SQL function (*basic\_example)*. And stop the session after done with all the jobs.



Run **./bin/spark-submit ./basic\_dataframe.py**

Sample output for line 18-20 (Job 5):

We get names of employees younger than 30 year-old as a result, which are Jenny and Tom. (The bottom lines of screenshot below)



**Exercise:**

Create your own dataset in json/txt format, reading it in and write it as parquet file saved under Spark directory, and then create a view from reading the parquet file you just created, lastly do 5 interesting queries on the SQL view. (Solution can be found in Github link)

**What is parquet?**

[Parquet](http://parquet.io/) is a columnar format that is supported by many other data processing systems. Spark SQL provides support for both reading and writing Parquet files that automatically preserves the schema of the original data.

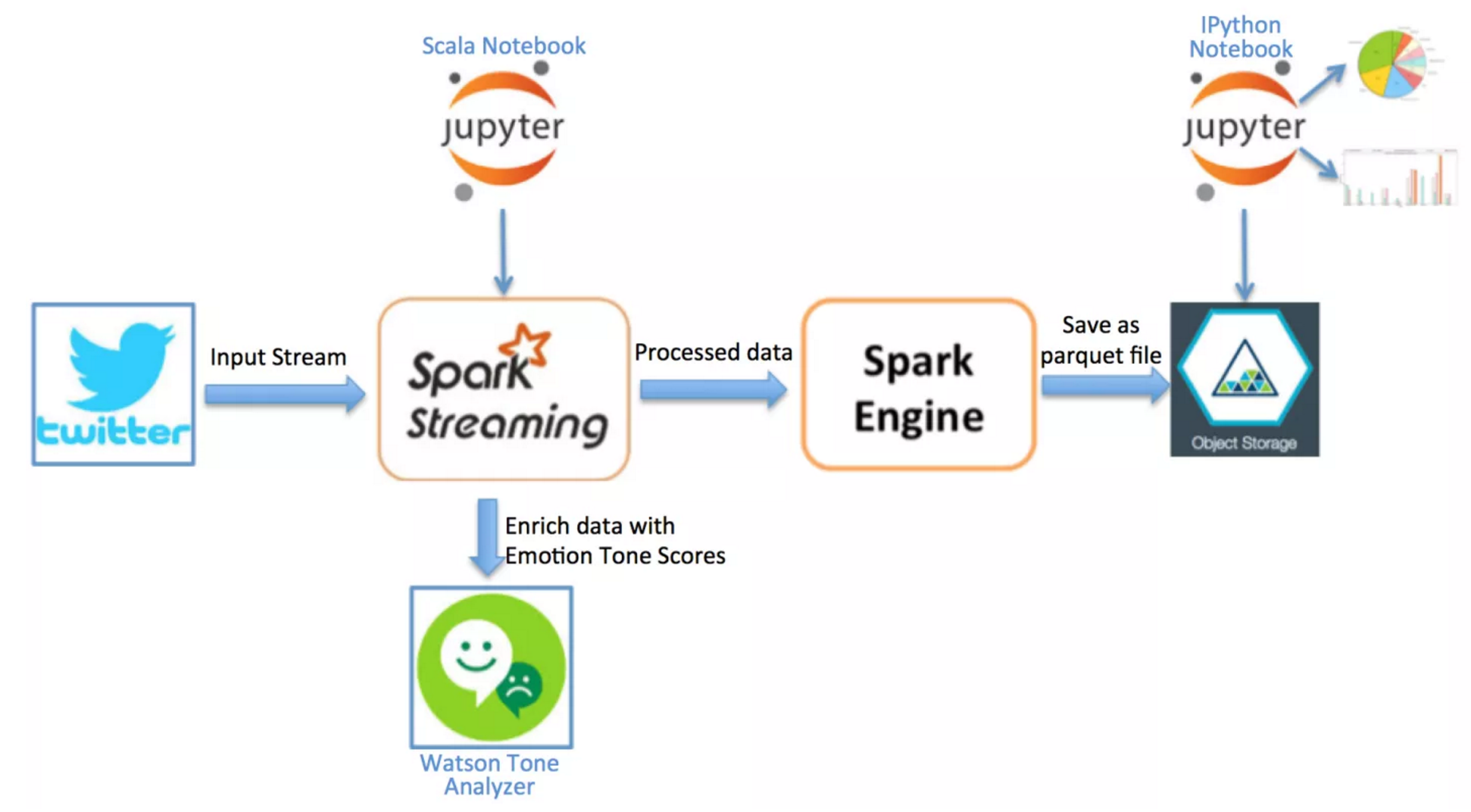
1. **Spark Streaming:**

* Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.
* Data can be ingested from many sources like Kafka, Flume, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window.
* Finally, processed data can be pushed out to filesystems, databases, and live dashboards.
* In fact, you can apply Spark’s [machine learning](http://spark.apache.org/docs/latest/ml-guide.html) and [graph processing](http://spark.apache.org/docs/latest/graphx-programming-guide.html) algorithms on data streams.
* **Spark Streaming Programming Guides:** <http://spark.apache.org/docs/latest/streaming-programming-guide.html>

**Example:**

IBM’s project (Sentiment Analysis of Twitter Hashtags): <https://developer.ibm.com/clouddataservices/sentiment-analysis-of-twitter-hashtags/>

They implemented a twitter stream fetching tool with Scala, and passed it through a sentiment analyzer to process the data. You can also write your own twitter stream fetching program in any other language, such as JAVA or Python. Spark engine will process the data and save it as parquet file, then you can analyze the data as you want. An example could be querying twitter message’s location with Spark SQL and plotting them on a map with different tags associated with their sentiment category.



1. **Spark for Machine Learning:**

MLlib is Spark’s scalable machine learning library. It consists of common algorithms like:

* Classification
* Regression
* Collaborative filtering
* K-means filtering
* Decomposition

Python and R are most popular languages used by data scientists. Spark provides data engineers and data scientists with a powerful, unified engine that is both fast and easy to use. This allows programmers to solve their machine learning problems interactively and at much greater scale.

In this tutorial, we have used Python to illustrate basic machine learning algorithms.   
We have used Tic Tac DataSet from UCLA repository to predict with the result as 1 (positive) or 0 (negative) for a given board configuration.  
<https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame>

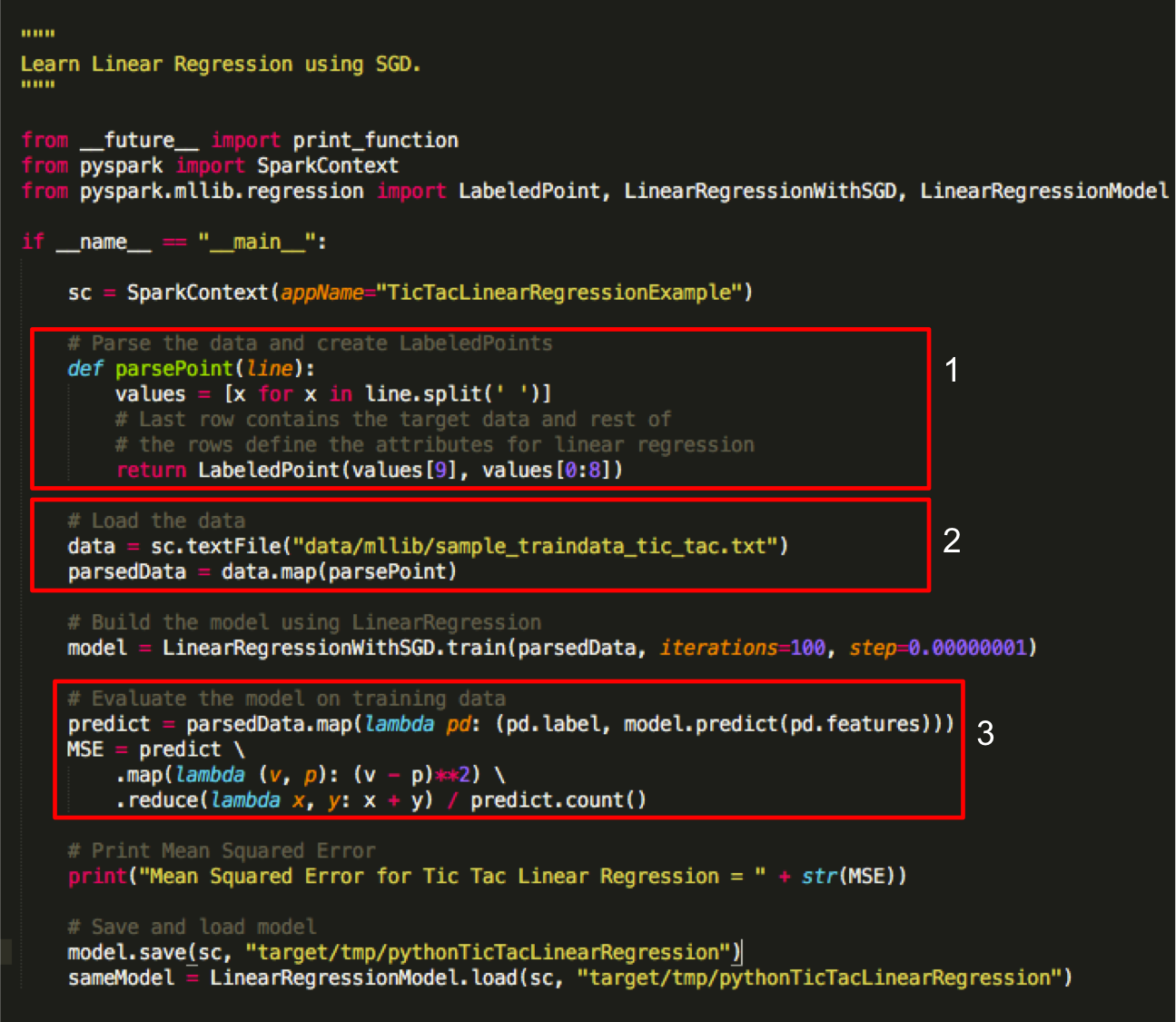
We have modified the data set as:

X - 1 (cross)

O - 0 (circle)

B - -1 (blank)

The code snippet describes the usage of Linear Regression using SGD (Stochastic Gradient Descent) to predict the outcome for Tic Tac Toe Configurations



Here, we have described the code flow:

1. Method parsePoint transforms the text to RDD [LabeledPoint]. A labeled point is a data set that is associated with a label or response.
2. The method parsePoint is called, mapped to the data and stored in parsedData.
3. Model is defined after performing linear regression and then applied transformation(map) and action(reduce).

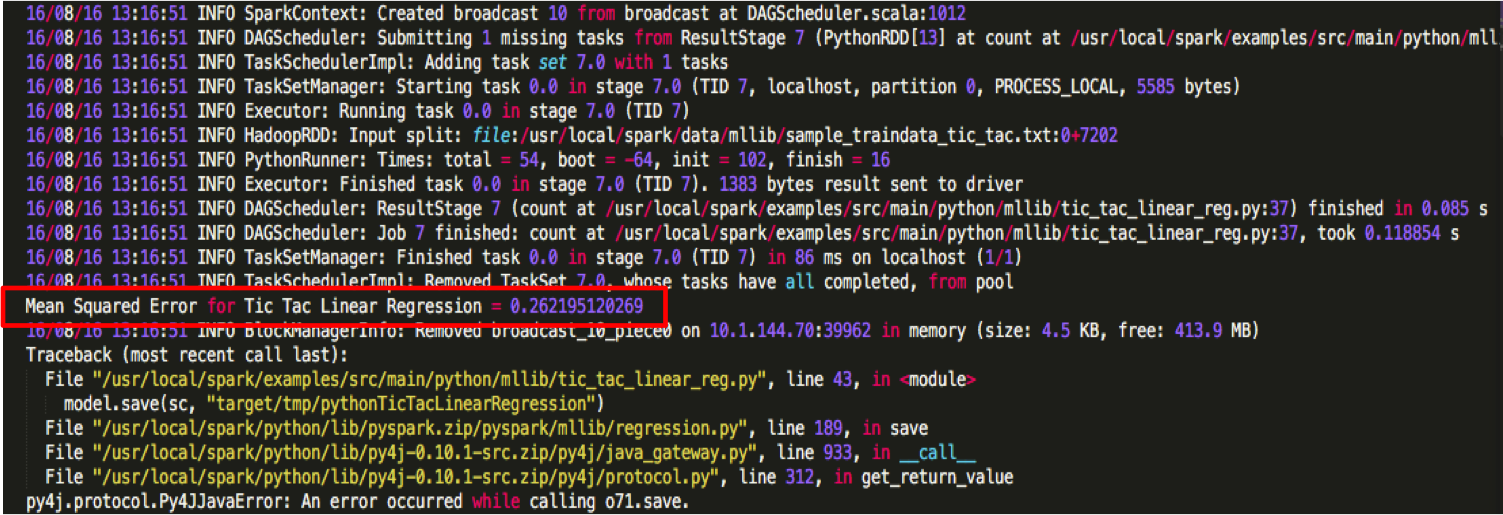
**How to run:**

Goto /usr/local/spark folder, run the following command:

$**./bin/spark-submit tic\_tac\_linear\_reg.py**

**Results:**

The below snippet is the outcome of running the above program. The MSE is highlighted in the red textbox.



**Exercise:**

Explore a new machine learning algorithm, SVD (Singular Value Decomposition) to run the above program.

The above program gives you an experience of running Python program using Spark. This will should be curious to solve more challenging real world problem using machine learning algorithms in Spark.

**What you learnt:**

Step-by-step demonstration:

1. What is Spark - its architecture.
2. Why use it
3. Run an example code
4. Write your own code snippet using Python example
5. Using SparkSQL
6. Using MLlib
7. Debugging and Tips

**References:**

<https://spark.apache.org/downloads.html>

<http://genomegeek.blogspot.com/2014/11/how-to-install-apache-spark-on-mac-os-x.html>

<https://spark.apache.org/docs/latest/graphx-programming-guide.html>

<https://aws.amazon.com/emr/details/spark/>

<https://aws.amazon.com/blogs/aws/new-apache-spark-on-amazon-emr/>

<http://pages.cs.wisc.edu/~akella/CS838/F15/assignment1.html>