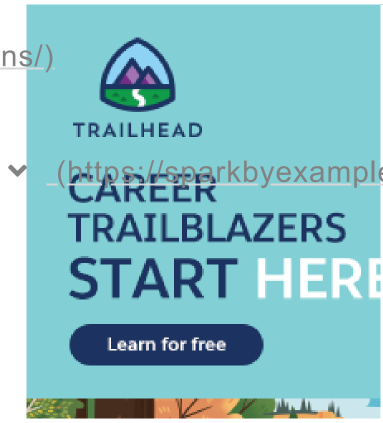
Spark (https://sparkbyexamples.com/)
PySpark (https://sparkbyexamples.com/pyspark-tutorial/)
Hive (https://sparkbyexamples.com/apache-hive-tutorial/)
HBase (https://sparkbyexamples.com/apache-hbase-tutorial/)
Kafka (https://sparkbyexamples.com/apache-kafka-tutorials-with-examples/)
FAQ's (https://sparkbyexamples.com/spark-questions/)
More (https://sparkbyexamples.com/)

Spark with Python (PySpark) Tutorial For Beginners

In this PySpark Tutorial (Spark with Python) with examples, you will learn what is PySpark? it's features, advantages, modules, packages, and how to use RDD & DataFrame with sample examples in Python code.



Every sample example explained here is tested in our development environment and is available at [PySpark Examples Github project (https://github.com/spark-examples/pyspark-examples/)](https://github.com/spark-examples/pyspark-examples/) for reference.

All Spark examples provided in this PySpark (Spark with Python) tutorial is basic, simple, and easy to practice for beginners who are enthusiastic to learn PySpark and advance your career in

PySpark – Accumulator (https://sparkbyexamples.com/pyspark/pyspark-accumulator-with-example/)

PySpark DataFrame

PySpark – Create a DataFrame (https://sparkbyexamples.com/pyspark/different-ways-to-create-dataframe-in-pyspark/)

PySpark – Create an empty DataFrame (https://sparkbyexamples.com/pyspark/pyspark-create-an-empty-dataframe/)

PySpark – Convert RDD to DataFrame (https://sparkbyexamples.com/pyspark/convert-pyspark-rdd-to-dataframe/)

PySpark – Convert DataFrame to Pandas (https://sparkbyexamples.com/pyspark/convert-pyspark-dataframe-to-pandas/)

PySpark – show() (https://sparkbyexamples.com/pyspark/pyspark-show-display-dataframe-contents-in-table/)

PySpark – StructType & StructField (https://sparkbyexamples.com/pyspark/pyspark-structtype-and-structfield/)

PySpark – Row Class (https://sparkbyexamples.com/pyspark/pyspark-row-using-rdd-dataframe/)

PySpark – Column Class (https://sparkbyexamples.com/pyspark/pyspark-column-functions/)

PySpark – select() (https://sparkbyexamples.com/pyspark/select-columns-from-pyspark-dataframe/)

PySpark – collect() (https://sparkbyexamples.com/pyspark/pyspark-collect/)

PySpark – withColumn() (https://sparkbyexamples.com/pyspark/pyspark-withcolumn/)

BigData and Machine Learning (https://en.wikipedia.org/wiki/Machine_learning).

Note: In case if you can't find the PySpark examples you are looking for on this tutorial page, I would recommend using the Search option from the menu bar to find your tutorial and sample example code, there are hundreds of tutorials in Spark, Scala, PySpark, and Python on this website you can learn from.

What is PySpark?

Before we jump into the PySpark tutorial, first, let's understand what is PySpark and how it is related to Python? who uses PySpark and it's advantages.

[PySpark – withColumnRenamed() (https://sparkbyexamples.com/pyspark/pyspark-rename-dataframe-column/)](https://sparkbyexamples.com/pyspark/pyspark-rename-dataframe-column/)

[PySpark – where() & filter() (https://sparkbyexamples.com/pyspark/pyspark-where-filter/)](https://sparkbyexamples.com/pyspark/pyspark-where-filter/)

[PySpark – drop() & dropDuplicates() (https://sparkbyexamples.com/pyspark/pyspark-distinct-to-drop-duplicates/)](https://sparkbyexamples.com/pyspark/pyspark-distinct-to-drop-duplicates/)

[PySpark – orderBy() and sort() (https://sparkbyexamples.com/pyspark/pyspark-orderby-and-sort-explained/)](https://sparkbyexamples.com/pyspark/pyspark-orderby-and-sort-explained/)

[PySpark – groupBy() (https://sparkbyexamples.com/pyspark/pyspark-groupby-explained-with-example/)](https://sparkbyexamples.com/pyspark/pyspark-groupby-explained-with-example/)

[PySpark – join() (https://sparkbyexamples.com/pyspark/pyspark-join-explained-with-examples/)](https://sparkbyexamples.com/pyspark/pyspark-join-explained-with-examples/)

[PySpark – union() & unionAll() (https://sparkbyexamples.com/pyspark/pyspark-union-and-unionall/)](https://sparkbyexamples.com/pyspark/pyspark-union-and-unionall/)

[PySpark – unionByName() (https://sparkbyexamples.com/spark/spark-merge-two-dataframes-with-different-columns/)](https://sparkbyexamples.com/spark/spark-merge-two-dataframes-with-different-columns/)

[PySpark – UDF (User Defined Function) (https://sparkbyexamples.com/pyspark/pyspark-udf-user-defined-function/)](https://sparkbyexamples.com/pyspark/pyspark-udf-user-defined-function/)

[PySpark – map() (https://sparkbyexamples.com/pyspark/pyspark-map-transformation/)](https://sparkbyexamples.com/pyspark/pyspark-map-transformation/)

[PySpark – flatMap() (https://sparkbyexamples.com/pyspark/pyspark-flatmap-transformation/)](https://sparkbyexamples.com/pyspark/pyspark-flatmap-transformation/)

[pyspark – foreach() (https://sparkbyexamples.com/pyspark/pyspark-loop-iterate-through-rows-in-dataframe/#use-foreach-loop-through-dataframe)](https://sparkbyexamples.com/pyspark/pyspark-loop-iterate-through-rows-in-dataframe/#use-foreach-loop-through-dataframe)

PySpark – sample() vs sampleBy()

Introduction

PySpark is a Spark library written in Python to run Python application using Apache Spark capabilities, using PySpark we can run applications parallelly on the distributed cluster (multiple nodes).

In other words, PySpark is a Python API for Apache Spark. Apache Spark is an analytical processing engine for large scale powerful distributed data processing and machine learning applications.

source: [https://databricks.com/ (https://databricks.com/)](https://databricks.com/)

Spark basically written in Scala and later on due to its industry adaptation it's API PySpark released for Python using Py4J. `Py4J` is a Java library that is integrated within PySpark and allows python to dynamically interface with JVM objects, hence to run PySpark you also need Java to be installed along with Python, and Apache Spark.

Additionally, For the development, you can use [Anaconda distribution (https://www.anaconda.com/)](https://www.anaconda.com/) (widely used in the Machine Learning community) which comes with a lot of useful tools like [Spyder IDE (https://www.spyder-ide.org/)](https://www.spyder-ide.org/), [Jupyter notebook (https://jupyter.org/)](https://jupyter.org/) to run PySpark applications.

(https://sparkbyexamples.com/pyspark/pyspark-sampling-example/)

PySpark – fillna() & fill() (https://sparkbyexamples.com/pyspark/pyspark-fillna-fill-replace-null-values/)

PySpark – pivot() (Row to Column) (https://sparkbyexamples.com/pyspark/pyspark-pivot-and-unpivot-dataframe/)

PySpark – partitionBy() (https://sparkbyexamples.com/pyspark/pyspark-partitionby-example/)

PySpark – ArrayType Column (Array) (https://sparkbyexamples.com/pyspark/pyspark-arraytype-column-with-examples/)

PySpark – MapType (Map/Dict) (https://sparkbyexamples.com/pyspark/pyspark-maptype-dict-examples/)

PySpark SQL Functions

PySpark – Aggregate Functions (https://sparkbyexamples.com/pyspark/pyspark-aggregate-functions/)

PySpark – Window Functions (https://sparkbyexamples.com/pyspark/pyspark-window-functions/)

PySpark – Date and Timestamp Functions (https://sparkbyexamples.com/pyspark/pyspark-sql-date-and-timestamp-functions/)

PySpark – JSON Functions (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/)

PySpark Datasources

PySpark – Read & Write CSV File (https://sparkbyexamples.com/pyspark/pyspark-read-csv-file-into-dataframe/)

PySpark – Read & Write Parquet File (https://sparkbyexamples.com/

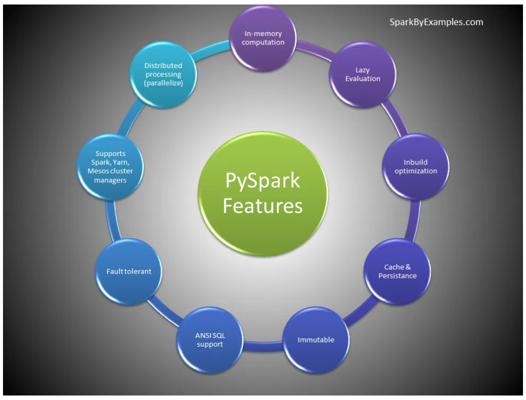
In real-time, PySpark has used a lot in the machine learning & Data scientists community; thanks to vast python machine learning libraries. Spark runs operations on billions and trillions of data on distributed clusters 100 times faster than the traditional python applications.

Who uses PySpark?

PySpark is very well used in Data Science and Machine Learning community as there are many widely used data science libraries written in Python including NumPy, TensorFlow also used due to its efficient processing of large datasets. PySpark has been used by many organizations like Walmart, Trivago, Sanofi, Runtastic, and many more.

Features

Following are the main features of PySpark.



PySpark Features

- In-memory computation
- Distributed processing using parallelize

[pyspark/pyspark-read-and-write-parquet-file/)](pyspark/pyspark-read-and-write-parquet-file/)

[PySpark – Read & Write JSON file (https://sparkbyexamples.com/pyspark/pyspark-read-json-file-into-dataframe/)](https://sparkbyexamples.com/pyspark/pyspark-read-json-file-into-dataframe/)

PySpark Built-In Functions

[PySpark – when() (https://sparkbyexamples.com/pyspark/pyspark-when-otherwise/)](https://sparkbyexamples.com/pyspark/pyspark-when-otherwise/)

[PySpark – expr() (https://sparkbyexamples.com/pyspark/pyspark-sql-expr-expression-function/)](https://sparkbyexamples.com/pyspark/pyspark-sql-expr-expression-function/)

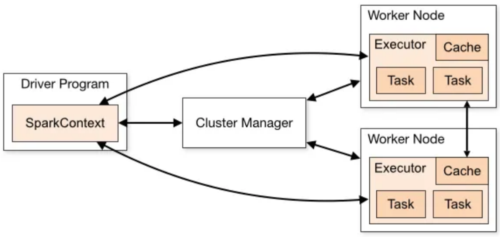
- Can be used with many cluster managers (Spark, Yarn, Mesos e.t.c)
- Fault-tolerant
- Immutable
- Lazy evaluation
- Cache & persistence
- Inbuild-optimization when using DataFrames
- upports ANSI SQL

vantages of PySpark

- ySpark is a general-purpose, in-iemory, distributed processing ngine that allows you to process ata efficiently in a distributed ashion.
- pplications running on PySpark are 100x faster than traditional systems.
- ou will get great benefits using ySpark for data ingestion pipelines.
- sing PySpark we can process data om Hadoop HDFS, AWS S3, and iany file systems.
- ySpark also is used to process real-me data using Streaming and Kafka.
- sing PySpark streaming you can lso stream files from the file system nd also stream from the socket.
- ySpark natively has machine earning and graph libraries.

Spark Architecture

che Spark works in a master-slave iitecture where the master is called ver” and slaves are called rkers”. When you run a Spark lication, Spark Driver creates a text that is an entry point to your lication, and all operations nsformations and actions) are cuted on worker nodes, and the resources are managed by Cluster Manager.



PySpark – lit() (https://sparkbyexamples.com/pyspark/pyspark-lit-add-literal-constant/)

PySpark – split() (https://sparkbyexamples.com/pyspark/pyspark-convert-string-to-array-column/)

PySpark – concat_ws() (https://sparkbyexamples.com/pyspark/pyspark-convert-array-column-to-string-column/)

Pyspark – substring() (https://sparkbyexamples.com/pyspark/pyspark-substring-from-a-column/)

PySpark – translate() (https://sparkbyexamples.com/pyspark/pyspark-replace-column-values/#translate-replace-character-by-character)

PySpark – regexp_replace() (https://sparkbyexamples.com/pyspark/pyspark-replace-column-values/#regexp_replace-replace-string-columns)

PySpark – overlay() (https://sparkbyexamples.com/pyspark/pyspark-replace-column-values/#overlay-function)

PySpark – to_timestamp() (https://sparkbyexamples.com/spark/pyspark-to_timestamp-convert-string-to-timestamp-type/)

PySpark – to_date() (https://sparkbyexamples.com/pyspark/pyspark-to_date-convert-timestamp-to-date/)

PySpark – date_format() (https://sparkbyexamples.com/pyspark/pyspark-date_format-convert-date-to-string-format/)

PySpark – datediff() (https://sparkbyexamples.com/pyspark/pyspark-difference-between-two-dates-days-months-years/#datediff)

PySpark – months_between() (https://sparkbyexamples.com/pyspark/pyspark-difference-between-two-dates-days-months-years/#months_between())

source: https://spark.apache.org/ (https://spark.apache.org/)

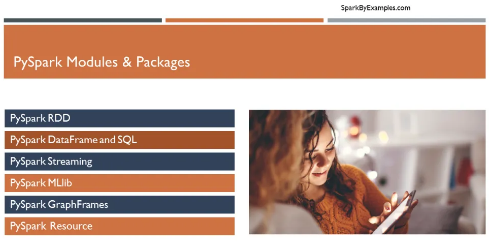
Cluster Manager Types

As of writing this Spark with Python (PySpark) tutorial, Spark supports below cluster managers:

- Standalone (https://spark.apache.org/docs/latest/spark-standalone.html) – a simple cluster manager included with Spark that makes it easy to set up a cluster.
- Apache Mesos (https://spark.apache.org/docs/latest/running-on-mesos.html) – Mesons is a Cluster manager that can also run Hadoop MapReduce and PySpark applications.
- Hadoop YARN (https://spark.apache.org/docs/latest/running-on-yarn.html) – the resource manager in Hadoop 2. This is mostly used, cluster manager.
- Kubernetes (https://spark.apache.org/docs/latest/running-on-kubernetes.html) – an open-source system for automating deployment, scaling, and management of containerized applications.

local – which is not really a cluster manager but still I wanted to mention as we use “local” for `master()` in order to run Spark on your laptop/computer.

PySpark Modules & Packages


SparkByExamples.com

PySpark Modules & Packages

PySpark RDD
PySpark DataFrame and SQL
PySpark Streaming
PySpark MLlib
PySpark GraphFrames
PySpark Resource

Modules & packages

- PySpark RDD (pyspark.RDD (https://spark.apache.org/docs/latest/

PySpark – explode() (https://sparkbyexamples.com/pyspark/pyspark-explode-nested-array-into-rows/)

PySpark – array_contains() (https://sparkbyexamples.com/pyspark/pyspark-arraytype-column-with-examples/#array_contains)

PySpark – array() (https://sparkbyexamples.com/pyspark/pyspark-arraytype-column-with-examples/#array)

PySpark – collect_list() (https://sparkbyexamples.com/pyspark/pyspark-aggregate-functions/#collect-list)

PySpark – collect_set() (https://sparkbyexamples.com/pyspark/pyspark-aggregate-functions/#collect-set)

PySpark – create_map() (https://sparkbyexamples.com/pyspark/pyspark-convert-dataframe-columns-to-maptype-dict/)

PySpark – map_keys() (https://sparkbyexamples.com/pyspark/pyspark-maptype-dict-examples/#map_keys)

PySpark – map_values() (https://sparkbyexamples.com/pyspark/pyspark-maptype-dict-examples/#map_values)

PySpark – struct() (https://sparkbyexamples.com/pyspark/pyspark-structtype-and-structfield/#update-struct-function)

PySpark – countDistinct() (https://sparkbyexamples.com/pyspark/pyspark-count-distinct-from-dataframe/)

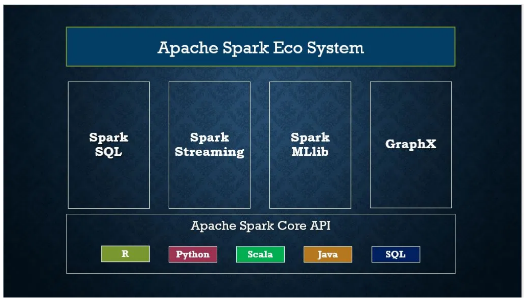
PySpark – sum(), avg() (https://sparkbyexamples.com/pyspark/pyspark-dataframe-groupby-and-sort-by-descending-order/)

PySpark – row_number() (https://sparkbyexamples.com/pyspark/pyspark-window-functions/#row_number)

PySpark – rank() (https://sparkbyexamples.com/pyspark/pyspark-window-functions/#rank)

api/python/pyspark.html#pyspark.RDD))

- PySpark DataFrame and SQL (pyspark.sql (https://spark.apache.org/docs/latest/api/python/pyspark.sql.html))
- PySpark Streaming (pyspark.streaming (https://spark.apache.org/docs/latest/api/python/pyspark.streaming.html))
- PySpark MLib (pyspark.ml (https://spark.apache.org/docs/latest/api/python/pyspark.ml.html), pyspark.mllib (https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html))
- PySpark GraphFrames (GraphFrames (https://graphframes.github.io/graphframes/docs/_site/index.html))
- PySpark Resource (pyspark.resource (https://spark.apache.org/docs/latest/api/python/pyspark.resource.html)) It’s new in PySpark 3.0



Besides these, if you wanted to use third-party libraries, you can find them at https://spark-packages.org/ (https://spark-packages.org/) . This page is kind of a repository of all Spark third-party libraries.

PySpark Installation

In order to run PySpark examples mentioned in this tutorial, you need to have Python, Spark and it’s needed tools to be installed on your computer. Since most developers use Windows for development, I will explain how to install PySpark on windows.

Install Python or Anaconda distribution

PySpark – dense_rank() (https://sparkbyexamples.com/pyspark/pyspark-window-functions/#dense_rank)

PySpark – percent_rank() (https://sparkbyexamples.com/pyspark/pyspark-window-functions/#percent_rank)

PySpark – typedLit() (https://sparkbyexamples.com/pyspark/pyspark-lit-add-literal-constant/#typedlit)

PySpark – from_json() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#from_json)

PySpark – to_json() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#to_json)

PySpark – json_tuple() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#json_tuple)

PySpark – get_json_object() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#get_json_object)

PySpark – schema_of_json() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#schema_of_json)

Download and install either Python from Python.org (https://www.python.org/downloads/windows/) or Anaconda distribution (https://www.anaconda.com/) which includes Python, Spyder IDE, and Jupyter notebook. I would recommend using Anaconda as it's popular and used by the Machine Learning & Data science community.

Install Java 8

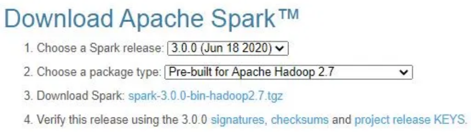
To run PySpark application, you would need Java 8 or later version hence download the Java version from Oracle (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) and install it on your system.

Post installation, set `JAVA_HOME` and `PATH` variable.

```
JAVA_HOME = C:\Program Files\Ja
PATH = %PATH%;C:\Program Files\
```

Install Apache Spark

Download Apache spark by accessing Spark Download (https://spark.apache.org/downloads.html) page and select the link from "Download Spark (point 3)". If you wanted to use a different version of Spark & Hadoop, select the one you wanted from drop downs and the link on point 3 changes to the selected version and provides you with an updated link to download.



Download Apache Spark™

1. Choose a Spark release: 3.0.0 (Jun 18 2020)
2. Choose a package type: Pre-built for Apache Hadoop 2.7
3. Download Spark: spark-3.0.0-bin-hadoop2.7.tgz
4. Verify this release using the 3.0.0 signatures, checksums and project release KEYS.

After download, untar the binary using 7zip (https://www.7-zip.org/download.html) and copy the underlying folder `spark-3.0.0-bin-hadoop2.7` to `c:\apps`

Now set the following environment variables.

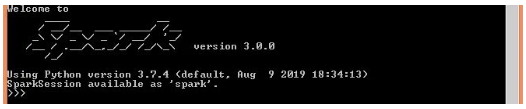
```
SPARK_HOME  = C:\apps\spark-3.0
HADOOP_HOME = C:\apps\spark-3.0
PATH=%PATH%;C:\apps\spark-3.0.0
```

Setup winutils.exe

Download wunutils.exe file from winutils (https://github.com/steveloughran/winutils/blob/master/hadoop-2.7.1/bin/winutils.exe), and copy it to `%SPARK_HOME%\bin` folder. Winutils are different for each Hadoop version hence download the right version from https://github.com/steveloughran/winutils (https://github.com/steveloughran/winutils)

PySpark shell

Now open command prompt and type `pyspark` command to run PySpark shell. You should see something like below.

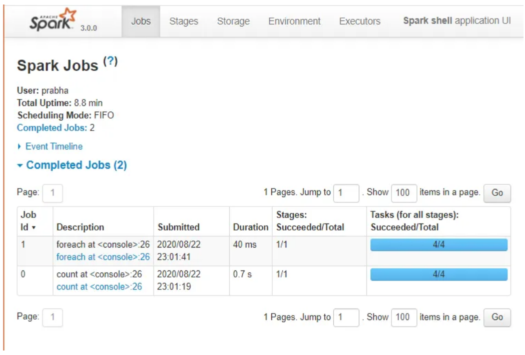


```
Welcome to
                                    version 3.0.0

Using Python version 3.7.4 (default, Aug  9 2019 18:34:13)
SparkSession available as 'spark'.
>>>
```

Spark-shell also creates a Spark context web UI (https://sparkbyexamples.com/spark/spark-web-ui-understanding/) and by default, it can access from http://localhost:4041 (http://localhost:4041).

Spark Web UI

Apache Spark provides a suite of Web UIs (Jobs, Stages, Tasks, Storage, Environment, Executors, and SQL) to monitor the status of your Spark application (https://sparkbyexamples.com/spark/spark-web-ui-understanding/), resource consumption of Spark cluster, and Spark configurations. On Spark Web UI, you can see how the operations are executed (https://sparkbyexamples.com/spark/spark-web-ui-understanding/).



Spark Jobs (?)

User: prabha
Total Uptime: 8.8 min
Scheduling Mode: FIFO
Completed Jobs: 2

▸ Event Timeline

▾ Completed Jobs (2)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	foreach at <console>:26 foreach at <console>:26	2020/08/22 23:01:41	40 ms	1/1	4/4
0	count at <console>:26 count at <console>:26	2020/08/22 23:01:19	0.7 s	1/1	4/4

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Spark Web UI

Spark History Server

Spark History servers, keep a log of all Spark application you submit by spark-submit (https://sparkbyexamples.com/spark/spark-submit-command/), spark-shell. before you start, first you need to set the below config on `spark-defaults.conf`

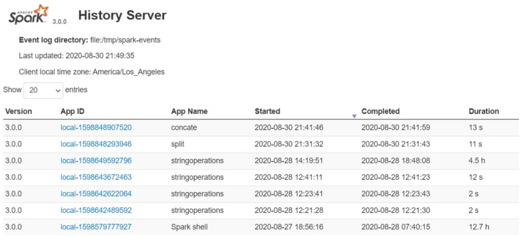
```
spark.eventLog.enabled true
spark.history.fs.logDirectory f
```

Now, start spark history server (https://sparkbyexamples.com/spark/spark-history-server-to-monitor-applications/) on Linux or mac by running.

```
$SPARK_HOME/sbin/start-history-
```

If you are running Spark on windows, you can start the history server by starting the below command.

```
$SPARK_HOME/bin/spark-class.cmd
```



Event log directory: file:/tmp/spark-events

Last updated: 2020-08-30 21:49:35

Client local time zone: America/Los_Angeles

Show 20 entries

Version	App ID	App Name	Started	Completed	Duration
3.0.0	local-1598848907520	concate	2020-08-30 21:41:46	2020-08-30 21:41:59	13 s
3.0.0	local-1598848293946	split	2020-08-30 21:31:32	2020-08-30 21:31:43	11 s
3.0.0	local-1598649592796	stringoperations	2020-08-28 14:19:51	2020-08-28 18:48:08	4.5 h
3.0.0	local-1598643672463	stringoperations	2020-08-28 12:41:11	2020-08-28 12:41:23	12 s
3.0.0	local-1598642622064	stringoperations	2020-08-28 12:23:41	2020-08-28 12:23:43	2 s
3.0.0	local-1598642489592	stringoperations	2020-08-28 12:21:28	2020-08-28 12:21:30	2 s
3.0.0	local-1598579777927	Spark shell	2020-08-27 18:56:16	2020-08-28 07:40:15	12.7 h

Spark History Server

By clicking on each App ID, you will get the details of the application in PySpark web UI (https://sparkbyexamples.com/spark/spark-web-ui-understanding/).

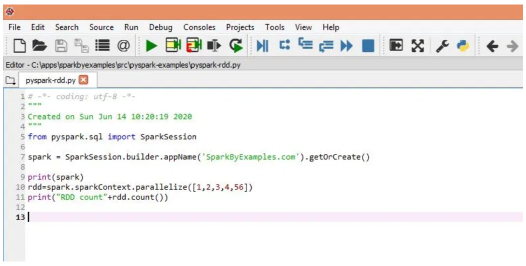
Spyder IDE & Jupyter Notebook

To write PySpark applications, you would need an IDE, there are 10's of IDE to work with and I choose to use Spyder IDE (https://sparkbyexamples.com/pyspark/setup-and-run-pyspark-on-spyder-ide/) and Jupyter notebook. If you have not installed Spyder IDE and Jupyter notebook along with Anaconda distribution, install these before you proceed.

Now, set the following environment variable.

```
PYTHONPATH => %SPARK_HOME%/pyth
```

Now open Spyder IDE and create a new file with below simple PySpark program and run it. You should see 5 in output.

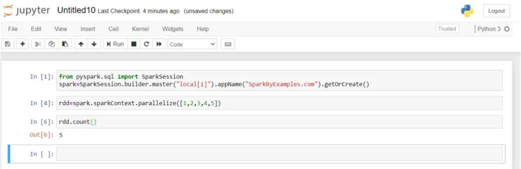


Editor - C:\apps\sparkbyexamples\src\pyspark-examples\pyspark-rdd.py

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 14 10:20:19 2020
"""
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

print(spark)
rdd=spark.sparkContext.parallelize([1,2,3,4,56])
print("RDD count"+rdd.count())

```

PySpark application running on Spyder IDE

Now let's start the Jupyter Notebook



```
In [1]: from pyspark.sql import SparkSession
        spark=SparkSession.builder.master("local[1]").appName("SparkByExamples.com").getOrCreate()

In [4]: rdd=spark.sparkContext.parallelize([1,2,3,4,5])

In [6]: rdd.count()
Out[6]: 5

In [ ]:
```

PySpark statements running on Jupyter Interface

PySpark RDD – Resilient Distributed Dataset

In this section of the PySpark tutorial, I will introduce the RDD and explains how to create them and use its transformation and action operations with examples. Here is the full article on PySpark RDD (https://sparkbyexamples.com/pyspark-rdd/) in case if you wanted to learn more of and get your fundamentals strong.

PySpark RDD (Resilient Distributed Dataset) (https://sparkbyexamples.com/pyspark-rdd/) is a fundamental data structure of PySpark that is fault-tolerant, immutable distributed collections of objects, which means once you create an RDD you cannot change it. Each dataset in RDD is divided into logical partitions, which can be computed on different nodes of the cluster.

RDD Creation

In order to create an RDD, first, you need to create a SparkSession which is an entry point to the PySpark application (https://sparkbyexamples.com/pyspark/pyspark-what-is-sparksession/). SparkSession can be created using a `builder()` or `newSession()` methods of the SparkSession.

Spark session internally creates a `sparkContext` variable of `SparkContext`. You can create multiple SparkSession objects but only one SparkContext per JVM. In case if you want to create another new

SparkContext you should stop existing Sparkcontext (using `stop()`) before creating a new one.

```
spark = SparkSession.builder()
      .master("local[1]")
      .appName("SparkByExamples
      .getOrCreate()
```

using parallelize()

SparkContext has several functions to use with RDDs. For example, it's `parallelize()` method is used to create an RDD from a list.

```
#Create RDD from parallelize
dataList = [("Java", 20000), ("
rdd=spark.sparkContext.parallel
```

using textFile()

RDD can also be created from a text file using `textFile()` function of the SparkContext.

```
//Create RDD from external Data
rdd2 = spark.sparkContext.textF
```

Once you have an RDD, you can perform transformation and action operations. Any operation you perform on RDD runs in parallel.

RDD Operations

On PySpark RDD, you can perform two kinds of operations.

RDD transformations – Transformations are lazy operations. When you run a transformation(for example update), instead of updating a current RDD, these operations return another RDD.

RDD actions – operations that trigger computation and return RDD values to the driver.

RDD Transformations

[Transformations on Spark RDD](https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-transformations/) (https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-transformations/) returns another RDD and transformations are lazy meaning they don't execute until you call an action on RDD. Some transformations on RDD's are `flatMap()`, `map()`, `reduceByKey()`, `filter()`, `sortByKey()` and return new RDD instead of updating the current.

RDD Actions

[RDD Action operation](https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-actions/) (https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-actions/) returns the values from an RDD to a driver node. In other words, any RDD function that returns non RDD[T] is considered as an action.

Some actions on RDD's are `count()`, `collect()`, `first()`, `max()`, `reduce()` and more.

PySpark DataFrame

DataFrame definition is very well explained by Databricks hence I do not want to define it again and confuse you. Below is the definition I took it from Databricks.

> *DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood. DataFrames can be constructed from a wide array of sources such as structured data files, tables in Hive, external databases, or existing RDDs.*
>
> — – Databricks

If you are coming from a Python background I would assume you already know what Pandas DataFrame is; PySpark DataFrame is mostly similar to Pandas DataFrame with exception PySpark DataFrames are distributed in the cluster (meaning the data in DataFrame's are stored in different machines in a cluster) and any operations in PySpark executes in parallel on all machines whereas Panda Dataframe stores and operates on a single machine.

If you have no Python background, I would recommend you learn some basics on Python before you proceeding this Spark tutorial. For now, just know that data in PySpark DataFrame's are stored in different machines in a cluster.

Is PySpark faster than pandas?

Due to parallel execution on all cores on multiple machines, Pyspark runs operations faster then Pandas. In other words, pandas run operations on a single node whereas PySpark runs on multiple machines.

DataFrame creation

Simplest way to create an DataFrame is from a Python list of data. DataFrame can also be created from an RDD and by reading a files from several sources.

using createDataFrame()

By using `createDataFrame()` function of the SparkSession you can create a DataFrame.

```
data = [('James','','Smith','19
  ('Michael','Rose','','2000-05
  ('Robert','','Williams','1978
  ('Maria','Anne','Jones','1967
  ('Jen','Mary','Brown','1980-0
]

columns = ["firstname","middlen
df = spark.createDataFrame(data
```

Since DataFrame's are structure format which contains names and column, we can get the schema of the DataFrame using `df.printSchema()`

```
data = [('James','','Smith','19
  ('Michael','Rose','','2000-05
  ('Robert','','Williams','1978
  ('Maria','Anne','Jones','1967
  ('Jen','Mary','Brown','1980-0
]

columns = ["firstname","middlen
df = spark.createDataFrame(data
```

`df.show()` shows the 20 elements from the DataFrame.

```
+---------+----------+--------+
|firstname|middlename|lastname|
+---------+----------+--------+
|James    |          |Smith   |
|Michael  |Rose      |        |
|Robert   |          |Williams|
|Maria    |Anne      |Jones   |
|Jen      |Mary      |Brown   |
+---------+----------+--------+
```

DataFrame operations

Like RDD, DataFrame also has operations like Transformations and Actions.

DataFrame from external data sources

In realtime applications, DataFrame's are created from external sources like files from the local system, HDFS, S3 Azure, HBase, MySQL table e.t.c. Below is an example of how to read a csv file from a local system.

```
df = spark.read.csv("/tmp/resou
df.printSchema()
```

Supported file formats

DataFrame has a rich set of API which supports reading and writing several file formats

- csv
- text
- Avro
- Parquet
- tsv
- xml and many more

DataFrame Examples

In this section of the PySpark Tutorial, you will find several Spark examples written in Python that help in your projects.

- Different ways to Create DataFrame in PySpark (https://sparkbyexamples.com/pyspark/different-ways-to-create-dataframe-in-pyspark/)
- PySpark – Ways to Rename column on DataFrame (https://sparkbyexamples.com/pyspark/pyspark-rename-dataframe-column/)
- PySpark withColumn() usage with Examples

(https://sparkbyexamples.com/pyspark/pyspark-dataframe-withcolumn/)

- PySpark – How to Filter data from DataFrame (https://sparkbyexamples.com/pyspark/pyspark-dataframe-filter/)
- PySpark orderBy() and sort() explained (https://sparkbyexamples.com/pyspark/pyspark-orderby-and-sort-explained/)
- PySpark explode array and map columns to rows (https://sparkbyexamples.com/pyspark/pyspark-explode-array-and-map-columns-to-rows/)
- PySpark – explode nested array into rows (https://sparkbyexamples.com/pyspark/pyspark-explode-nested-array-into-rows/)
- PySpark Read CSV file into DataFrame (https://sparkbyexamples.com/pyspark/pyspark-read-csv-file-into-dataframe/)
- PySpark Groupby Explained with Examples (https://sparkbyexamples.com/pyspark/pyspark-groupby-explained-with-example/)
- PySpark Aggregate Functions with Examples (https://sparkbyexamples.com/pyspark/pyspark-aggregate-functions/)
- PySpark Joins Explained with Examples (https://sparkbyexamples.com/pyspark/pyspark-join-explained-with-examples/)

PySpark SQL Tutorial

`PySpark SQL` is one of the most used Py**Spark** modules which is used for processing structured columnar data format. Once you have a DataFrame created, you can interact with the data by using SQL syntax.

In other words, Spark SQL brings native RAW SQL queries on Spark meaning you can run traditional ANSI SQL's on Spark Dataframe, in the later section of this PySpark SQL tutorial, you will learn in details using SQL `select`, `where`, `group by`, `join`, `union` e.t.c

In order to use SQL, first, create a temporary table on DataFrame using `createOrReplaceTempView()` function. Once created, this table can be accessed throughout the SparkSession using `sql()` and it will be dropped along with your SparkContext termination.

Use `sql()` method of the SparkSession object to run the query and this method returns a new DataFrame.

```
df.createOrReplaceTempView("PERS
df2 = spark.sql("SELECT * from P
df2.printSchema()
df2.show()
```

Let's see another pyspark example using `group by`.

```
groupDF = spark.sql("SELECT gend
groupDF.show()
```

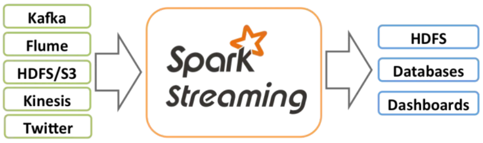
This yields the below output

```
+------+--------+
|gender|count(1)|
+------+--------+
|     F|       2|
|     M|       3|
+------+--------+
```

Similarly you can run any traditional SQL queries on DataFrame's using PySpark SQL.

PySpark Streaming Tutorial

PySpark Streaming is a scalable, high-throughput, fault-tolerant streaming processing system that supports both batch and streaming workloads. It is used to process real-time data from sources like file system folder, TCP socket, S3 (https://aws.amazon.com/s3/), Kafka (https://kafka.apache.org/), Flume (https://en.wikipedia.org/wiki/Apache_Flume), Twitter (https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data), and Amazon Kinesis (https://aws.amazon.com/kinesis/) to name a few. The processed data can be pushed to databases, Kafka, live dashboards e.t.c



source: https://spark.apache.org/ (https://spark.apache.org/)

Streaming from TCP Socket

Use `readStream.format("socket")` from Spark session object to read data from the socket and provide options host and port where you want to stream data from.

```
df = spark.readStream
      .format("socket")
      .option("host","localhost
      .option("port","9090")
      .load()
```

Spark reads the data from socket and represents it in a “value” column of DataFrame. `df.printSchema()` outputs

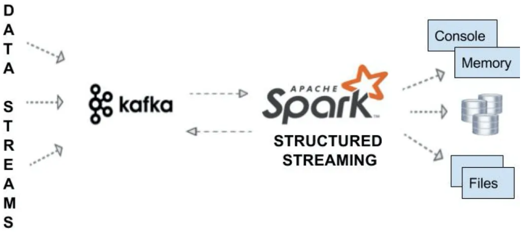
```
root
 |-- value: string (nullable = 
```

After processing, you can stream the DataFrame to console. In real-time, we ideally stream it to either Kafka, database e.t.c

```
query = count.writeStream
      .format("console")
      .outputMode("complete")
      .start()
      .awaitTermination()
```

Streaming from Kafka

Using Spark Streaming we can read from Kafka topic and write to Kafka (https://sparkbyexamples.com/spark/spark-streaming-from-kafka-topic/) topic in TEXT, CSV, AVRO (https://avro.apache.org/) and JSON formats



```
df = spark.readStream
        .format("kafka")
        .option("kafka.bootstra
        .option("subscribe", "j
        .option("startingOffset
        .load()
```

Below pyspark example, writes message to another topic in Kafka using `writeStream()`

```
df.selectExpr("CAST(id AS STRIN
    .writeStream
    .format("kafka")
    .outputMode("append")
    .option("kafka.bootstrap.ser
    .option("topic", "josn_data_
    .start()
    .awaitTermination()
```

PySpark MLlib

In this section, I will cover pyspark examples by using MLlib library.

PySpark GraphFrames

PySpark GraphFrames are introduced in Spark 3.0 (https://sparkbyexamples.com/spark/spark-3-0-features-with-examples-part-i/) version to support Graphs on DataFrame’s. Prior to 3.0, Spark has GraphX library which ideally runs on RDD and loses all Data Frame capabilities.

GraphFrames is a package for Apache Spark which provides DataFrame-based Graphs. It provides high-level APIs in Scala, Java, and Python. It aims to provide both the functionality of GraphX and extended functionality taking advantage of Spark DataFrames. This extended functionality includes motif finding, DataFrame-based serialization,

> *and highly expressive graph queries.*
>
> — – GRAPHFRAMES.GITHUB.IO (HTTPS://GRAPHFRAMES.GITHUB.IO/)

Difference between GraphX and GraphFrame

GraphX works on RDDs where as GraphFrames works with DataFrames.

References

Below are some of the articles/tutorials I've referred.

- https://spark.apache.org/docs/latest/api/python/pyspark.html (https://spark.apache.org/docs/latest/api/python/pyspark.html)
- https://spark.apache.org/docs/latest/rdd-programming-guide.html (https://spark.apache.org/docs/latest/rdd-programming-guide.html)

Share this:

(https://sparkbyexamples.com/pyspark-tutorial/?share=facebook&nb=1)

(https://sparkbyexamples.com/pyspark-tutorial/?share=reddit&nb=1)

(https://sparkbyexamples.com/pyspark-tutorial/?share=pinterest&nb=1) 1

(https://sparkbyexamples.com/pyspark-tutorial/?share=tumblr&nb=1)

(https://sparkbyexamples.com/pyspark-tutorial/?share=pocket&nb=1)

(https://sparkbyexamples.com/pyspark-tutorial/?share=linkedin&nb=1)

(https://sparkbyexamples.com/pyspark-tutorial/?share=twitter&nb=1)

Categories

Apache Hadoop (https://sparkbyexamples.com/category/hadoop/)

Apache Spark (https://sparkbyexamples.com/category/spark/)

Apache Spark Streaming (https://sparkbyexamples.com/category/spark/apache-spark-streaming/)

Apache Kafka (https://sparkbyexamples.com/category/kafka/)

Apache HBase (https://sparkbyexamples.com/category/hbase/)

Apache Cassandra (https://sparkbyexamples.com/category/cassandra/)

Snowflake Database (https://sparkbyexamples.com/category/snowflake/)

H2O Sparkling Water (https://sparkbyexamples.com/category/h2o-sparkling-water/)

PySpark (https://sparkbyexamples.com/category/pyspark/)

Recent Posts

Spark regexp_replace() – Replace String Value (https://sparkbyexamples.com/spark/spark-regexp_replace-replace-string-value/)

How to Run a PySpark Script from Python? (https://sparkbyexamples.com/pyspark/run-pyspark-script-from-python-subprocess/)

Spark SQL like() Using Wildcard Example (https://sparkbyexamples.com/spark/spark-sql-like-using-wildcard-example/)

Spark isin() & IS NOT IN Operator Example (https://sparkbyexamples.com/spark/spark-isin-is-not-in-operator-example/)

Spark – Get Size/Length of Array & Map Column (https://sparkbyexamples.com/spark/spark-get-size-length-of-array-map-column/)

Spark Using Length/Size Of a DataFrame Column (https://sparkbyexamples.com/spark/spark-using-length-size-of-a-dataframe-column/)

Spark rlike() Working with Regex Matching Examples (https://sparkbyexamples.com/spark/spark-rlike-regex-matching-examples/)

Spark Check String Column Has Numeric Values (https://sparkbyexamples.com/spark/spark-check-string-column-has-numeric-values/)

Spark Check Column Data Type is Integer or String (https://sparkbyexamples.com/spark/spark-check-column-data-type-is-integer-or-string/)

About SparkByExamples.Com

SparkByExamples.com is a Big Data and Spark examples community page, all examples are simple and easy to understand, and well tested in our development environment Read more .. (https://sparkbyexamples.com/about-sparkbyexamples/)

Follow Us

(https://twitter.com/sparkbyexamples)

(https://www.facebook.com/sparkbyexamples/)

(https://www.linkedin.com/in/n-nk-b860a8193/)

(https://github.com/spark-examples/)