PySpark Cheat Sheet: Spark in Python

March 21st, 2017 in Python

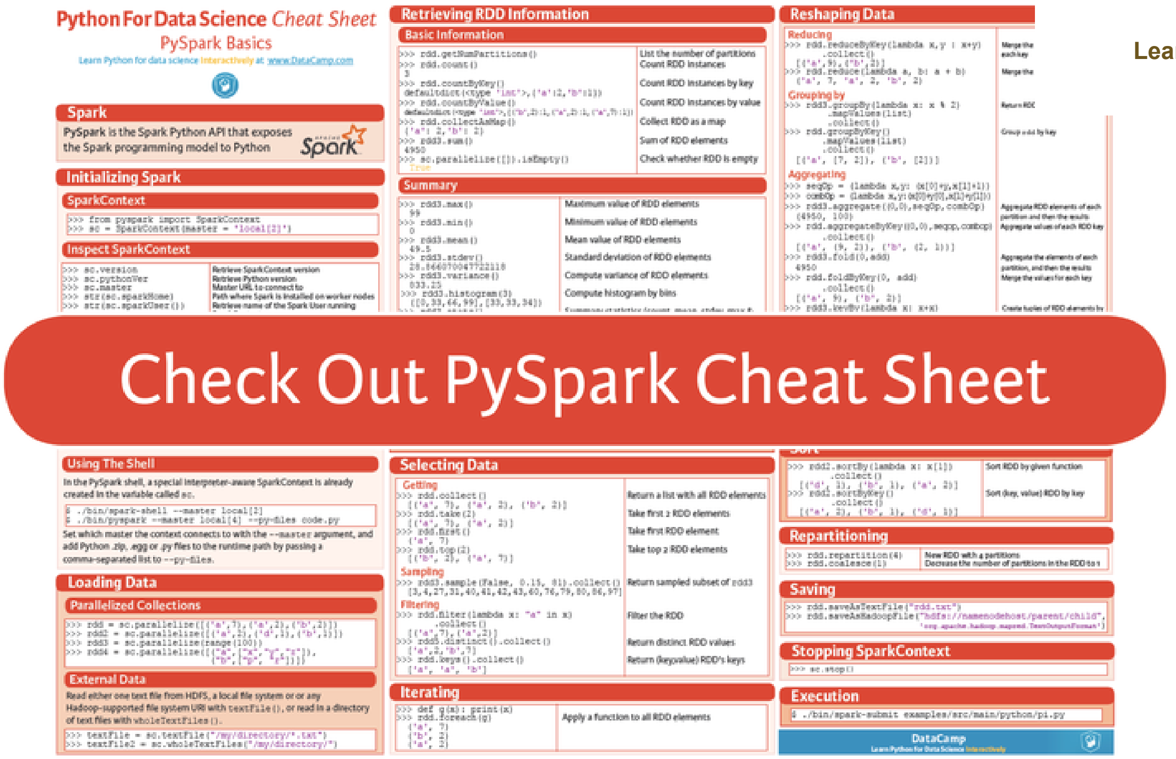
Karlijn Willems

Apache Spark is generally known as a fast, general and open-source engine for big data processing, with built-in modules for streaming, SQL, machine learning and graph processing. It allows you to speed analytic applications up to 100 times faster compared to technologies on the market today. You can interface Spark with Python through "PySpark". This is the Spark Python API exposes the Spark programming model to Python.

Even though working with Spark will remind you in many ways of working with Pandas DataFrames, you'll also see that it can be tough getting familiar with all the functions that you can use to query, transform, inspect, ... your data. What's more, if you've never worked with any other programming language or if you're new to the field, it might be hard to distinguish between RDD operations.

learners and start one of our interactive tutorials today!

Learn R **Learn Python**



Python For Data Science *Cheat Sheet*

PySpark Basics

Learn Python for data science Interactively at www.DataCamp.com

Spark

PySpark is the Spark Python API that exposes the Spark programming model to Python

Initializing Spark

SparkContext

```
>>> from pyspark import SparkContext
>>> sc = SparkContext(master = 'local[2]')
```

Inspect SparkContext

`>>> sc.version`	Retrieve SparkContext version
`>>> sc.pythonVer`	Retrieve Python version
`>>> sc.master`	Master URL to connect to
`>>> str(sc.sparkHome)`	Path where Spark is installed on worker nodes
`>>> str(sc.sparkUser())`	Retrieve name of the Spark User running

Retrieving RDD Information

Basic Information

`>>> rdd.getNumPartitions()`	List the number of partitions

`>>> rdd.count()`
`3`
	Count RDD instances

`>>> rdd.countByKey()`
`defaultdict(<type 'int'>,{'a':2,'b':1})`
	Count RDD instances by key

`>>> rdd.countByValue()`
`defaultdict(<type 'int'>,{('b',2):1,('a',2):1,('a',7):1})`
	Count RDD instances by value

`>>> rdd.collectAsMap()`
`{'a': 2,'b': 2}`
	Collect RDD as a map

`>>> rdd3.sum()`
`4950`
	Sum of RDD elements

`>>> sc.parallelize([]).isEmpty()`
`True`
	Check whether RDD is empty

Summary

`>>> rdd3.max()`
`99`
	Maximum value of RDD elements

`>>> rdd3.min()`
`0`
	Minimum value of RDD elements

`>>> rdd3.mean()`
`49.5`
	Mean value of RDD elements

`>>> rdd3.stdev()`
`28.866070047722118`
	Standard deviation of RDD elements

`>>> rdd3.variance()`
`833.25`
	Compute variance of RDD elements

`>>> rdd3.histogram(3)`
`([0,33,66,99],[33,33,34])`
	Compute histogram by bins

Reshaping Data

Reducing

`>>> rdd.reduceByKey(lambda x,y : x+y)`
`.collect()`
`[('a',9),('b',2)]`
	Merge the each key

`>>> rdd.reduce(lambda a, b: a + b)`
`('a', 7, 'a', 2, 'b', 2)`
	Merge the

Grouping by

`>>> rdd3.groupBy(lambda x: x % 2)`
`.mapValues(list)`
`.collect()`
	Return RDD

`>>> rdd.groupByKey()`
`.mapValues(list)`
`.collect()`
`[('a', [7, 2]), ('b', [2])]`
	Group rdd by key

Aggregating

`>>> seqOp = (lambda x,y: (x[0]+y,x[1]+1))`
`>>> combOp = (lambda x,y:(x[0]+y[0],x[1]+y[1]))`
`>>> rdd3.aggregate((0,0),seqOp,combOp)`
`(4950, 100)`
	Aggregate RDD elements of each partition and then the results

`>>> rdd.aggregateByKey((0,0),seqop,combop)`
`.collect()`
`[('a', (9, 2)), ('b', (2, 1))]`
	Aggregate values of each RDD key

`>>> rdd3.fold(0,add)`
`4950`
	Aggregate the elements of each partition, and then the results

`>>> rdd.foldByKey(0, add)`
`.collect()`
`[('a', 9), ('b', 2)]`
	Merge the values for each key

`>>> rdd3.keyBy(lambda x: x+x)`	Create tuples of RDD elements by

Check Out PySpark Cheat Sheet

Using The Shell

In the PySpark shell, a special interpreter-aware SparkContext is already created in the variable called sc.

```
$ ./bin/spark-shell --master local[2]
$ ./bin/pyspark --master local[4] --py-files code.py
```

Set which master the context connects to with the `--master` argument, and add Python .zip, .egg or .py files to the runtime path by passing a comma-separated list to `--py-files`.

Loading Data

Parallelized Collections

```
>>> rdd = sc.parallelize([('a',7),('a',2),('b',2)])
>>> rdd2 = sc.parallelize([('a',2),('d',1),('b',1)])
>>> rdd3 = sc.parallelize(range(100))
>>> rdd4 = sc.parallelize([("a",["x","y","z"]),
                           ("b",["p", "r"])])
```

External Data

Read either one text file from HDFS, a local file system or or any Hadoop-supported file system URI with `textFile()`, or read in a directory of text files with `wholeTextFiles()`.

```
>>> textFile = sc.textFile("/my/directory/*.txt")
>>> textFile2 = sc.wholeTextFiles("/my/directory/")
```

Selecting Data

Getting

`>>> rdd.collect()`
`[('a', 7), ('a', 2), ('b', 2)]`
	Return a list with all RDD elements

`>>> rdd.take(2)`
`[('a', 7), ('a', 2)]`
	Take first 2 RDD elements

`>>> rdd.first()`
`('a', 7)`
	Take first RDD element

`>>> rdd.top(2)`
`[('b', 2), ('a', 7)]`
	Take top 2 RDD elements

Sampling

`>>> rdd3.sample(False, 0.15, 81).collect()`
`[3,4,27,31,40,41,42,43,60,76,79,80,86,97]`
	Return sampled subset of `rdd3`

Filtering

`>>> rdd.filter(lambda x: "a" in x)`
`.collect()`
`[('a',7),('a',2)]`
	Filter the RDD

`>>> rdd5.distinct().collect()`
`['a',2,'b',7]`
	Return distinct RDD values

`>>> rdd.keys().collect()`
`['a', 'a', 'b']`
	Return (key,value) RDD's keys

Iterating

`>>> def g(x): print(x)`
`>>> rdd.foreach(g)`
`('a', 7)`
`('b', 2)`
`('a', 2)`
	Apply a function to all RDD elements

Sort

`>>> rdd2.sortBy(lambda x: x[1])`
`.collect()`
`[('d', 1), ('b', 1), ('a', 2)]`
	Sort RDD by given function

`>>> rdd2.sortByKey()`
`.collect()`
`[('a', 2), ('b', 1), ('d', 1)]`
	Sort (key, value) RDD by key

Repartitioning

`>>> rdd.repartition(4)`	New RDD with 4 partitions
`>>> rdd.coalesce(1)`	Decrease the number of partitions in the RDD to 1

Saving

```
>>> rdd.saveAsTextFile("rdd.txt")
>>> rdd.saveAsHadoopFile("hdfs://namenodehost/parent/child",
                         'org.apache.hadoop.mapred.TextOutputFormat')
```

Stopping SparkContext

```
>>> sc.stop()
```

Execution

```
$ ./bin/spark-submit examples/src/main/python/pi.py
```

DataCamp
Learn Python for Data Science Interactively

Even though the documentation is very elaborate, it never hurts to have a cheat sheet by your side, especially when you're just getting into it.

This PySpark cheat sheet covers the basics, from initializing Spark and loading your data, to retrieving RDD information, sorting, filtering and sampling your data. But that's not all. You'll also see that topics such as repartitioning, iterating, merging, saving your data and stopping the SparkContext are included in the cheat sheet.

Are you hungry for more? Don't miss our other Python cheat sheets for data science that cover top basics, Numpy, Pandas, Pandas Data Wrangling and much more!

What do you think?

Python

learners and start one of our interactive tutorials today!

Learn R Learn Python

Up Next

New Course! Supervised Learning in R: Classification

R Programming 79 views

This beginner-level introduction to machine learning covers four of the most common classification algorithms. You will come away with a b…

September 27th, 2017 in Blog

New Course: Case Studies in Statistical Thinking!

Python

1,139 views

Hone your applied data science skills in Python by doing a variety of case studies across multiple disciplines. Use data science to solve …

September 20th, 2017 in Blog

Jupyter Notebook Cheat Sheet

Python

19,148 views

This Jupyter Notebook cheat sheet will help you to find your way around the well-known Jupyter Notebook App, a subproject of Project Jupyt…

September 19th, 2017 in Blog

Comments

vaidas-armonas

Hi Karlijn, thanks for sharing this. But *why* an RDD cheat sheet? RDDs should be left for the most exotic of uses while most DataSet APIs where we get Tungsten performance benefits and more familiar interface. I would love to hear the rationale and to share with the analysts in my company :)

06/28/17 6:43 AM |

karlijn

Hi vaidas-armonas! Thanks for your message! The idea behind this is that when you start out with Spark, you usually first cover the basic building blocks, which are still the RDDs, even though you might not use them often in daily practice. I hope this reasoning makes somewhat sense to you; You might also have seen that I recently added a cheat sheet to work with Spark SQL (DataFrames) on the community. You can find it here: https://www.datacamp.com/community/blog/pyspark-sql-cheat-sheet :)

07/26/17 7:45 AM |

vaidas-armonas

Hi Karlijn, thanks for sharing this. But *why* an RDD cheat sheet? RDDs should be left for the most exotic of uses while most use cases should be covered by DataFrame / DataSet APIs where we get Tungsten performance benefits and more familiar interface. I would love to hear the rationale and maybe a cheat sheet for PySpark DataFrames to share with the analysts in my company :)

06/28/17 6:42 AM |

karlijn

Hi vaidas-armonas, I replied to your message above :)

07/26/17 7:47 AM |

alfredo-g-marquez

This is great! Can't wait to see the cheat sheet for SparkR.

03/24/17 4:57 AM |

benmainye

I have never used PySpark in my work. But, it seems awesome. Thanks for the cheatsheet.

03/22/17 4:23 AM |

karlijn

Hi there! Thanks for writing. Spark is really wonderful when you're working with big data. Definitely give PySpark a go and let us know what you thought of it!

03/22/17 7:49 AM |

mannurulz