|  |
| --- |
| PySpark Core Modules  • pyspark.SparkContext  • pyspark.RDD  • pyspark.Broadcast  • pyspark.Accumulator  • pyspark.AccumulatorParam  • pyspark.SparkConf  • pyspark.SparkFiles  • pyspark.StorageLevel  • pyspark.TaskContext  • pyspark.RDDBarrier  • pyspark.BarrierTaskContext  • pyspark.BarrierTaskInfo  • pyspark.InheritableThread  • pyspark.util.VersionUtils |

**pyspark.SparkContext**

pyspark.SparkContext(master=None, appName=None, sparkHome=None, pyFiles=None, environment=None, batchSize=0, serializer=PickleSerializer(), conf=None, gateway=None, jsc=None, profiler\_cls=<class 'pyspark.profiler.BasicProfiler'>)

Main entry point for Spark functionality. A SparkContext represents the connection to a Spark cluster, and can be used to create RDD and broadcast variables on that cluster.

When you create a new SparkContext, at least the master and app name should be set, either through the named parameters here or through conf.

Only one SparkContext should be active per JVM. You must stop() the active SparkContext before creating a new one.

SparkContext instance is not supported to share across multiple processes out of the box, and PySpark does not guarantee multi-processing execution. Use threads instead for concurrent processing purpose.

|  |
| --- |
| **Parameters:**  **master: *str, optional***  Cluster URL to connect to (e.g. mesos://host:port, spark://host:port, local[4]).  **appName: *str, optional***  A name for your job, to display on the cluster web UI.  **sparkHome: *str, optional***  Location where Spark is installed on cluster nodes.  **pyFiles: *list, optional***  Collection of .zip or .py files to send to the cluster and add to PYTHONPATH. These can be paths on the local file system or HDFS, HTTP, HTTPS, or FTP URLs.  **environment: *dict, optional***  A dictionary of environment variables to set on worker nodes.  **batchSize: *int, optional***  The number of Python objects represented as a single Java object. Set 1 to disable batching, 0 to automatically choose the batch size based on object sizes, or -1 to use an unlimited batch size  **serializer: *pyspark.serializers.Serializer, optional***  The serializer for RDDs.  **conf:** [***pyspark.SparkConf***](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.html#pyspark.SparkConf)***, optional***  An object setting Spark properties.  **gateway: *py4j.java\_gateway.JavaGateway, optional***  Use an existing gateway and JVM, otherwise a new JVM will be instantiated. This is only used internally.  **Jsc: *py4j.java\_gateway.JavaObject, optional***  The JavaSparkContext instance. This is only used internally.  **profiler\_cls: *type, optional***  A class of custom Profiler used to do profiling (default is **pyspark.profiler.BasicProfiler**). |
| **Methods**   |  |  | | --- | --- | | [**accumulator**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.accumulator.html#pyspark.SparkContext.accumulator)(value[, accum\_param]) | Create an [**Accumulator**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Accumulator.html#pyspark.Accumulator) with the given initial value, using a given [**AccumulatorParam**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.AccumulatorParam.html#pyspark.AccumulatorParam) helper object to define how to add values of the data type if provided. | | [**addFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.addFile.html#pyspark.SparkContext.addFile)(path[, recursive]) | Add a file to be downloaded with this Spark job on every node. | | [**addPyFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.addPyFile.html#pyspark.SparkContext.addPyFile)(path) | Add a .py or .zip dependency for all tasks to be executed on this SparkContext in the future. | | [**binaryFiles**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.binaryFiles.html#pyspark.SparkContext.binaryFiles)(path[, minPartitions]) | Read a directory of binary files from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI as a byte array. | | [**binaryRecords**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.binaryRecords.html#pyspark.SparkContext.binaryRecords)(path, recordLength) | Load data from a flat binary file, assuming each record is a set of numbers with the specified numerical format (see ByteBuffer), and the number of bytes per record is constant. | | [**broadcast**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.broadcast.html#pyspark.SparkContext.broadcast)(value) | Broadcast a read-only variable to the cluster, returning a [**Broadcast**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Broadcast.html#pyspark.Broadcast) object for reading it in distributed functions. | | [**cancelAllJobs**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.cancelAllJobs.html#pyspark.SparkContext.cancelAllJobs)() | Cancel all jobs that have been scheduled or are running. | | [**cancelJobGroup**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.cancelJobGroup.html#pyspark.SparkContext.cancelJobGroup)(groupId) | Cancel active jobs for the specified group. | | [**dump\_profiles**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.dump_profiles.html#pyspark.SparkContext.dump_profiles)(path) | Dump the profile stats into directory *path* | | [**emptyRDD**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.emptyRDD.html#pyspark.SparkContext.emptyRDD)() | Create an RDD that has no partitions or elements. | | [**getCheckpointDir**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.getCheckpointDir.html#pyspark.SparkContext.getCheckpointDir)() | Return the directory where RDDs are checkpointed. | | [**getConf**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.getConf.html#pyspark.SparkContext.getConf)() |  | | [**getLocalProperty**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.getLocalProperty.html#pyspark.SparkContext.getLocalProperty)(key) | Get a local property set in this thread, or null if it is missing. | | [**getOrCreate**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.getOrCreate.html#pyspark.SparkContext.getOrCreate)([conf]) | Get or instantiate a SparkContext and register it as a singleton object. | | [**hadoopFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.hadoopFile.html#pyspark.SparkContext.hadoopFile)(path, inputFormatClass, keyClass, …) | Read an ‘old’ Hadoop InputFormat with arbitrary key and value class from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI. | | [**hadoopRDD**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.hadoopRDD.html#pyspark.SparkContext.hadoopRDD)(inputFormatClass, keyClass, valueClass) | Read an ‘old’ Hadoop InputFormat with arbitrary key and value class, from an arbitrary Hadoop configuration, which is passed in as a Python dict. | | [**newAPIHadoopFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.newAPIHadoopFile.html#pyspark.SparkContext.newAPIHadoopFile)(path, inputFormatClass, …) | Read a ‘new API’ Hadoop InputFormat with arbitrary key and value class from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI. | | [**newAPIHadoopRDD**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.newAPIHadoopRDD.html#pyspark.SparkContext.newAPIHadoopRDD)(inputFormatClass, keyClass, …) | Read a ‘new API’ Hadoop InputFormat with arbitrary key and value class, from an arbitrary Hadoop configuration, which is passed in as a Python dict. | | [**parallelize**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.parallelize.html#pyspark.SparkContext.parallelize)(c[, numSlices]) | Distribute a local Python collection to form an RDD. | | [**pickleFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.pickleFile.html#pyspark.SparkContext.pickleFile)(name[, minPartitions]) | Load an RDD previously saved using [**RDD.saveAsPickleFile()**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsPickleFile.html#pyspark.RDD.saveAsPickleFile) method. | | [**range**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.range.html#pyspark.SparkContext.range)(start[, end, step, numSlices]) | Create a new RDD of int containing elements from *start* to *end* (exclusive), increased by *step* every element. | | [**runJob**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.runJob.html#pyspark.SparkContext.runJob)(rdd, partitionFunc[, partitions, …]) | Executes the given partitionFunc on the specified set of partitions, returning the result as an array of elements. | | [**sequenceFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.sequenceFile.html#pyspark.SparkContext.sequenceFile)(path[, keyClass, valueClass, …]) | Read a Hadoop SequenceFile with arbitrary key and value Writable class from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI. | | [**setCheckpointDir**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.setCheckpointDir.html#pyspark.SparkContext.setCheckpointDir)(dirName) | Set the directory under which RDDs are going to be checkpointed. | | [**setJobDescription**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.setJobDescription.html#pyspark.SparkContext.setJobDescription)(value) | Set a human readable description of the current job. | | [**setJobGroup**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.setJobGroup.html#pyspark.SparkContext.setJobGroup)(groupId, description[, …]) | Assigns a group ID to all the jobs started by this thread until the group ID is set to a different value or cleared. | | [**setLocalProperty**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.setLocalProperty.html#pyspark.SparkContext.setLocalProperty)(key, value) | Set a local property that affects jobs submitted from this thread, such as the Spark fair scheduler pool. | | [**setLogLevel**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.setLogLevel.html#pyspark.SparkContext.setLogLevel)(logLevel) | Control our logLevel. | | [**setSystemProperty**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.setSystemProperty.html#pyspark.SparkContext.setSystemProperty)(key, value) | Set a Java system property, such as spark.executor.memory. | | [**show\_profiles**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.show_profiles.html#pyspark.SparkContext.show_profiles)() | Print the profile stats to stdout | | [**sparkUser**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.sparkUser.html#pyspark.SparkContext.sparkUser)() | Get SPARK\_USER for user who is running SparkContext. | | [**statusTracker**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.statusTracker.html#pyspark.SparkContext.statusTracker)() | Return **StatusTracker** object | | [**stop**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.stop.html#pyspark.SparkContext.stop)() | Shut down the SparkContext. | | [**textFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.textFile.html#pyspark.SparkContext.textFile)(name[, minPartitions, use\_unicode]) | Read a text file from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI, and return it as an RDD of Strings. | | [**union**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.union.html#pyspark.SparkContext.union)(rdds) | Build the union of a list of RDDs. | | [**wholeTextFiles**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.wholeTextFiles.html#pyspark.SparkContext.wholeTextFiles)(path[, minPartitions, …]) | Read a directory of text files from HDFS, a local file system (available on all nodes), or any Hadoop-supported file system URI. | |
| **Attributes**   |  |  | | --- | --- | | [**PACKAGE\_EXTENSIONS**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.PACKAGE_EXTENSIONS.html#pyspark.SparkContext.PACKAGE_EXTENSIONS) |  | | [**applicationId**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.applicationId.html#pyspark.SparkContext.applicationId) | A unique identifier for the Spark application. | | [**defaultMinPartitions**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.defaultMinPartitions.html#pyspark.SparkContext.defaultMinPartitions) | Default min number of partitions for Hadoop RDDs when not given by user | | [**defaultParallelism**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.defaultParallelism.html#pyspark.SparkContext.defaultParallelism) | Default level of parallelism to use when not given by user (e.g. | | [**resources**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.resources.html#pyspark.SparkContext.resources) |  | | [**startTime**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.startTime.html#pyspark.SparkContext.startTime) | Return the epoch time when the Spark Context was started. | | [**uiWebUrl**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.uiWebUrl.html#pyspark.SparkContext.uiWebUrl) | Return the URL of the SparkUI instance started by this SparkContext | | [**version**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.version.html#pyspark.SparkContext.version) | The version of Spark on which this application is running. | |
| **from** **pyspark.context** **import** SparkContext  **>>>** sc = SparkContext('local', 'test')  **>>>** sc2 = SparkContext('local', 'test2')  Traceback (most recent call last):  ...  ValueError: ... |

**pyspark.RDD**

pyspark.RDD(jrdd, ctx, jrdd\_deserializer=AutoBatchedSerializer(PickleSerializer()))

A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. Represents an immutable, partitioned collection of elements that can be operated on in parallel.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Methods**   |  |  | | --- | --- | | [**aggregate**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.aggregate.html#pyspark.RDD.aggregate)(zeroValue, seqOp, combOp) | Aggregate the elements of each partition, and then the results for all the partitions, using a given combine functions and a neutral “zero value.” | | [**aggregateByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.aggregateByKey.html#pyspark.RDD.aggregateByKey)(zeroValue, seqFunc, combFunc) | Aggregate the values of each key, using given combine functions and a neutral “zero value”. | | [**barrier**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.barrier.html#pyspark.RDD.barrier)() | Marks the current stage as a barrier stage, where Spark must launch all tasks together. | | [**cache**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.cache.html#pyspark.RDD.cache)() | Persist this RDD with the default storage level (*MEMORY\_ONLY*). | | [**cartesian**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.cartesian.html#pyspark.RDD.cartesian)(other) | Return the Cartesian product of this RDD and another one, that is, the RDD of all pairs of elements (a, b) where a is in *self* and b is in *other*. | | [**checkpoint**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.checkpoint.html#pyspark.RDD.checkpoint)() | Mark this RDD for checkpointing. | | [**coalesce**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.coalesce.html#pyspark.RDD.coalesce)(numPartitions[, shuffle]) | Return a new RDD that is reduced into *numPartitions* partitions. | | [**cogroup**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.cogroup.html#pyspark.RDD.cogroup)(other[, numPartitions]) | For each key k in *self* or *other*, return a resulting RDD that contains a tuple with the list of values for that key in *self* as well as *other*. | | [**collect**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.collect.html#pyspark.RDD.collect)() | Return a list that contains all of the elements in this RDD. | | [**collectAsMap**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.collectAsMap.html#pyspark.RDD.collectAsMap)() | Return the key-value pairs in this RDD to the master as a dictionary. | | [**collectWithJobGroup**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.collectWithJobGroup.html#pyspark.RDD.collectWithJobGroup)(groupId, description[, …]) | When collect rdd, use this method to specify job group. | | [**combineByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.combineByKey.html#pyspark.RDD.combineByKey)(createCombiner, mergeValue, …) | Generic function to combine the elements for each key using a custom set of aggregation functions. | | [**count**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.count.html#pyspark.RDD.count)() | Return the number of elements in this RDD. | | [**countApprox**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.countApprox.html#pyspark.RDD.countApprox)(timeout[, confidence]) | Approximate version of count() that returns a potentially incomplete result within a timeout, even if not all tasks have finished. | | [**countApproxDistinct**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.countApproxDistinct.html#pyspark.RDD.countApproxDistinct)([relativeSD]) | Return approximate number of distinct elements in the RDD. | | [**countByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.countByKey.html#pyspark.RDD.countByKey)() | Count the number of elements for each key, and return the result to the master as a dictionary. | | [**countByValue**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.countByValue.html#pyspark.RDD.countByValue)() | Return the count of each unique value in this RDD as a dictionary of (value, count) pairs. | | [**distinct**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.distinct.html#pyspark.RDD.distinct)([numPartitions]) | Return a new RDD containing the distinct elements in this RDD. | | [**filter**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.filter.html#pyspark.RDD.filter)(f) | Return a new RDD containing only the elements that satisfy a predicate. | | [**first**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.first.html#pyspark.RDD.first)() | Return the first element in this RDD. | | [**flatMap**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.flatMap.html#pyspark.RDD.flatMap)(f[, preservesPartitioning]) | Return a new RDD by first applying a function to all elements of this RDD, and then flattening the results. | | [**flatMapValues**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.flatMapValues.html#pyspark.RDD.flatMapValues)(f) | Pass each value in the key-value pair RDD through a flatMap function without changing the keys; this also retains the original RDD’s partitioning. | | [**fold**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.fold.html#pyspark.RDD.fold)(zeroValue, op) | Aggregate the elements of each partition, and then the results for all the partitions, using a given associative function and a neutral “zero value.” | | [**foldByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.foldByKey.html#pyspark.RDD.foldByKey)(zeroValue, func[, numPartitions, …]) | Merge the values for each key using an associative function “func” and a neutral “zeroValue” which may be added to the result an arbitrary number of times, and must not change the result (e.g., 0 for addition, or 1 for multiplication.). | | [**foreach**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.foreach.html#pyspark.RDD.foreach)(f) | Applies a function to all elements of this RDD. | | [**foreachPartition**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.foreachPartition.html#pyspark.RDD.foreachPartition)(f) | Applies a function to each partition of this RDD. | | [**fullOuterJoin**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.fullOuterJoin.html#pyspark.RDD.fullOuterJoin)(other[, numPartitions]) | Perform a right outer join of *self* and *other*. | | [**getCheckpointFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.getCheckpointFile.html#pyspark.RDD.getCheckpointFile)() | Gets the name of the file to which this RDD was checkpointed | | [**getNumPartitions**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.getNumPartitions.html#pyspark.RDD.getNumPartitions)() | Returns the number of partitions in RDD | | [**getResourceProfile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.getResourceProfile.html#pyspark.RDD.getResourceProfile)() | Get the [**pyspark.resource.ResourceProfile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.resource.ResourceProfile.html#pyspark.resource.ResourceProfile) specified with this RDD or None if it wasn’t specified. | | [**getStorageLevel**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.getStorageLevel.html#pyspark.RDD.getStorageLevel)() | Get the RDD’s current storage level. | | [**glom**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.glom.html#pyspark.RDD.glom)() | Return an RDD created by coalescing all elements within each partition into a list. | | [**groupBy**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.groupBy.html#pyspark.RDD.groupBy)(f[, numPartitions, partitionFunc]) | Return an RDD of grouped items. | | [**groupByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.groupByKey.html#pyspark.RDD.groupByKey)([numPartitions, partitionFunc]) | Group the values for each key in the RDD into a single sequence. | | [**groupWith**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.groupWith.html#pyspark.RDD.groupWith)(other, \*others) | Alias for cogroup but with support for multiple RDDs. | | [**histogram**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.histogram.html#pyspark.RDD.histogram)(buckets) | Compute a histogram using the provided buckets. | | [**id**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.id.html#pyspark.RDD.id)() | A unique ID for this RDD (within its SparkContext). | | [**intersection**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.intersection.html#pyspark.RDD.intersection)(other) | Return the intersection of this RDD and another one. | | [**isCheckpointed**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.isCheckpointed.html#pyspark.RDD.isCheckpointed)() | Return whether this RDD is checkpointed and materialized, either reliably or locally. | | [**isEmpty**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.isEmpty.html#pyspark.RDD.isEmpty)() | Returns true if and only if the RDD contains no elements at all. | | [**isLocallyCheckpointed**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.isLocallyCheckpointed.html#pyspark.RDD.isLocallyCheckpointed)() | Return whether this RDD is marked for local checkpointing. | | [**join**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.join.html#pyspark.RDD.join)(other[, numPartitions]) | Return an RDD containing all pairs of elements with matching keys in *self* and *other*. | | [**keyBy**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.keyBy.html#pyspark.RDD.keyBy)(f) | Creates tuples of the elements in this RDD by applying *f*. | | [**keys**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.keys.html#pyspark.RDD.keys)() | Return an RDD with the keys of each tuple. | | [**leftOuterJoin**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.leftOuterJoin.html#pyspark.RDD.leftOuterJoin)(other[, numPartitions]) | Perform a left outer join of *self* and *other*. | | [**localCheckpoint**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.localCheckpoint.html#pyspark.RDD.localCheckpoint)() | Mark this RDD for local checkpointing using Spark’s existing caching layer. | | [**lookup**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.lookup.html#pyspark.RDD.lookup)(key) | Return the list of values in the RDD for key *key*. | | [**map**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.map.html#pyspark.RDD.map)(f[, preservesPartitioning]) | Return a new RDD by applying a function to each element of this RDD. | | [**mapPartitions**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.mapPartitions.html#pyspark.RDD.mapPartitions)(f[, preservesPartitioning]) | Return a new RDD by applying a function to each partition of this RDD. | | [**mapPartitionsWithIndex**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.mapPartitionsWithIndex.html#pyspark.RDD.mapPartitionsWithIndex)(f[, …]) | Return a new RDD by applying a function to each partition of this RDD, while tracking the index of the original partition. | | [**mapPartitionsWithSplit**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.mapPartitionsWithSplit.html#pyspark.RDD.mapPartitionsWithSplit)(f[, …]) | Return a new RDD by applying a function to each partition of this RDD, while tracking the index of the original partition. | | [**mapValues**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.mapValues.html#pyspark.RDD.mapValues)(f) | Pass each value in the key-value pair RDD through a map function without changing the keys; this also retains the original RDD’s partitioning. | | [**max**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.max.html#pyspark.RDD.max)([key]) | Find the maximum item in this RDD. | | [**mean**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.mean.html#pyspark.RDD.mean)() | Compute the mean of this RDD’s elements. | | [**meanApprox**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.meanApprox.html#pyspark.RDD.meanApprox)(timeout[, confidence]) | Approximate operation to return the mean within a timeout or meet the confidence. | | [**min**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.min.html#pyspark.RDD.min)([key]) | Find the minimum item in this RDD. | | [**name**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.name.html#pyspark.RDD.name)() | Return the name of this RDD. | | [**partitionBy**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.partitionBy.html#pyspark.RDD.partitionBy)(numPartitions[, partitionFunc]) | Return a copy of the RDD partitioned using the specified partitioner. | | [**persist**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.persist.html#pyspark.RDD.persist)([storageLevel]) | Set this RDD’s storage level to persist its values across operations after the first time it is computed. | | [**pipe**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.pipe.html#pyspark.RDD.pipe)(command[, env, checkCode]) | Return an RDD created by piping elements to a forked external process. | | [**randomSplit**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.randomSplit.html#pyspark.RDD.randomSplit)(weights[, seed]) | Randomly splits this RDD with the provided weights. | | [**reduce**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.reduce.html#pyspark.RDD.reduce)(f) | Reduces the elements of this RDD using the specified commutative and associative binary operator. | | [**reduceByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.reduceByKey.html#pyspark.RDD.reduceByKey)(func[, numPartitions, partitionFunc]) | Merge the values for each key using an associative and commutative reduce function. | | [**reduceByKeyLocally**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.reduceByKeyLocally.html#pyspark.RDD.reduceByKeyLocally)(func) | Merge the values for each key using an associative and commutative reduce function, but return the results immediately to the master as a dictionary. | | [**repartition**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.repartition.html#pyspark.RDD.repartition)(numPartitions) | Return a new RDD that has exactly numPartitions partitions. | | [**repartitionAndSortWithinPartitions**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.repartitionAndSortWithinPartitions.html#pyspark.RDD.repartitionAndSortWithinPartitions)([…]) | Repartition the RDD according to the given partitioner and, within each resulting partition, sort records by their keys. | | [**rightOuterJoin**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.rightOuterJoin.html#pyspark.RDD.rightOuterJoin)(other[, numPartitions]) | Perform a right outer join of *self* and *other*. | | [**sample**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sample.html#pyspark.RDD.sample)(withReplacement, fraction[, seed]) | Return a sampled subset of this RDD. | | [**sampleByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sampleByKey.html#pyspark.RDD.sampleByKey)(withReplacement, fractions[, seed]) | Return a subset of this RDD sampled by key (via stratified sampling). | | [**sampleStdev**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sampleStdev.html#pyspark.RDD.sampleStdev)() | Compute the sample standard deviation of this RDD’s elements (which corrects for bias in estimating the standard deviation by dividing by N-1 instead of N). | | [**sampleVariance**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sampleVariance.html#pyspark.RDD.sampleVariance)() | Compute the sample variance of this RDD’s elements (which corrects for bias in estimating the variance by dividing by N-1 instead of N). | | [**saveAsHadoopDataset**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsHadoopDataset.html#pyspark.RDD.saveAsHadoopDataset)(conf[, keyConverter, …]) | Output a Python RDD of key-value pairs (of form RDD[(K, V)]) to any Hadoop file system, using the old Hadoop OutputFormat API (mapred package). | | [**saveAsHadoopFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsHadoopFile.html#pyspark.RDD.saveAsHadoopFile)(path, outputFormatClass[, …]) | Output a Python RDD of key-value pairs (of form RDD[(K, V)]) to any Hadoop file system, using the old Hadoop OutputFormat API (mapred package). | | [**saveAsNewAPIHadoopDataset**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsNewAPIHadoopDataset.html#pyspark.RDD.saveAsNewAPIHadoopDataset)(conf[, …]) | Output a Python RDD of key-value pairs (of form RDD[(K, V)]) to any Hadoop file system, using the new Hadoop OutputFormat API (mapreduce package). | | [**saveAsNewAPIHadoopFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsNewAPIHadoopFile.html#pyspark.RDD.saveAsNewAPIHadoopFile)(path, outputFormatClass) | Output a Python RDD of key-value pairs (of form RDD[(K, V)]) to any Hadoop file system, using the new Hadoop OutputFormat API (mapreduce package). | | [**saveAsPickleFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsPickleFile.html#pyspark.RDD.saveAsPickleFile)(path[, batchSize]) | Save this RDD as a SequenceFile of serialized objects. | | [**saveAsSequenceFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsSequenceFile.html#pyspark.RDD.saveAsSequenceFile)(path[, compressionCodecClass]) | Output a Python RDD of key-value pairs (of form RDD[(K, V)]) to any Hadoop file system, using the “org.apache.hadoop.io.Writable” types that we convert from the RDD’s key and value types. | | [**saveAsTextFile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.saveAsTextFile.html#pyspark.RDD.saveAsTextFile)(path[, compressionCodecClass]) | Save this RDD as a text file, using string representations of elements. | | [**setName**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.setName.html#pyspark.RDD.setName)(name) | Assign a name to this RDD. | | [**sortBy**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sortBy.html#pyspark.RDD.sortBy)(keyfunc[, ascending, numPartitions]) | Sorts this RDD by the given keyfunc | | [**sortByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sortByKey.html#pyspark.RDD.sortByKey)([ascending, numPartitions, keyfunc]) | Sorts this RDD, which is assumed to consist of (key, value) pairs. | | [**stats**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.stats.html#pyspark.RDD.stats)() | Return a **StatCounter** object that captures the mean, variance and count of the RDD’s elements in one operation. | | [**stdev**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.stdev.html#pyspark.RDD.stdev)() | Compute the standard deviation of this RDD’s elements. | | [**subtract**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.subtract.html#pyspark.RDD.subtract)(other[, numPartitions]) | Return each value in *self* that is not contained in *other*. | | [**subtractByKey**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.subtractByKey.html#pyspark.RDD.subtractByKey)(other[, numPartitions]) | Return each (key, value) pair in *self* that has no pair with matching key in *other*. | | [**sum**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sum.html#pyspark.RDD.sum)() | Add up the elements in this RDD. | | [**sumApprox**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.sumApprox.html#pyspark.RDD.sumApprox)(timeout[, confidence]) | Approximate operation to return the sum within a timeout or meet the confidence. | | [**take**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.take.html#pyspark.RDD.take)(num) | Take the first num elements of the RDD. | | [**takeOrdered**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.takeOrdered.html#pyspark.RDD.takeOrdered)(num[, key]) | Get the N elements from an RDD ordered in ascending order or as specified by the optional key function. | | [**takeSample**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.takeSample.html#pyspark.RDD.takeSample)(withReplacement, num[, seed]) | Return a fixed-size sampled subset of this RDD. | | [**toDebugString**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.toDebugString.html#pyspark.RDD.toDebugString)() | A description of this RDD and its recursive dependencies for debugging. | | [**toLocalIterator**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.toLocalIterator.html#pyspark.RDD.toLocalIterator)([prefetchPartitions]) | Return an iterator that contains all of the elements in this RDD. | | [**top**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.top.html#pyspark.RDD.top)(num[, key]) | Get the top N elements from an RDD. | | [**treeAggregate**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.treeAggregate.html#pyspark.RDD.treeAggregate)(zeroValue, seqOp, combOp[, depth]) | Aggregates the elements of this RDD in a multi-level tree pattern. | | [**treeReduce**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.treeReduce.html#pyspark.RDD.treeReduce)(f[, depth]) | Reduces the elements of this RDD in a multi-level tree pattern. | | [**union**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.union.html#pyspark.RDD.union)(other) | Return the union of this RDD and another one. | | [**unpersist**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.unpersist.html#pyspark.RDD.unpersist)([blocking]) | Mark the RDD as non-persistent, and remove all blocks for it from memory and disk. | | [**values**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.values.html#pyspark.RDD.values)() | Return an RDD with the values of each tuple. | | [**variance**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.variance.html#pyspark.RDD.variance)() | Compute the variance of this RDD’s elements. | | [**withResources**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.withResources.html#pyspark.RDD.withResources)(profile) | Specify a [**pyspark.resource.ResourceProfile**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.resource.ResourceProfile.html#pyspark.resource.ResourceProfile) to use when calculating this RDD. | | [**zip**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.zip.html#pyspark.RDD.zip)(other) | Zips this RDD with another one, returning key-value pairs with the first element in each RDD second element in each RDD, etc. | | [**zipWithIndex**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.zipWithIndex.html#pyspark.RDD.zipWithIndex)() | Zips this RDD with its element indices. | | [**zipWithUniqueId**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.zipWithUniqueId.html#pyspark.RDD.zipWithUniqueId)() | Zips this RDD with generated unique Long ids. | |
| **Attributes**   |  |  | | --- | --- | | [**context**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.context.html#pyspark.RDD.context) | The [**SparkContext**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkContext.html#pyspark.SparkContext) that this RDD was created on. | |

**pyspark.SparkConf**

pyspark.SparkConf(loadDefaults=True, \_jvm=None, \_jconf=None)

Configuration for a Spark application. Used to set various Spark parameters as key-value pairs.

Most of the time, you would create a SparkConf object with SparkConf(), which will load values from spark.\* Java system properties as well. In this case, any parameters you set directly on the SparkConf object take priority over system properties.

For unit tests, you can also call SparkConf(false) to skip loading external settings and get the same configuration no matter what the system properties are.

All setter methods in this class support chaining. For example, you can write conf.setMaster("local").setAppName("My app").

Once a SparkConf object is passed to Spark, it is cloned and can no longer be modified by the user.

|  |
| --- |
| **Parameters:**  **loadDefaults*bool***  whether to load values from Java system properties (True by default)  **\_jvm*class:py4j.java\_gateway.JVMView***  internal parameter used to pass a handle to the Java VM; does not need to be set by users  **\_jconf*class:py4j.java\_gateway.JavaObject***  Optionally pass in an existing SparkConf handle to use its parameters |
| **Methods**   |  |  | | --- | --- | | [**contains**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.contains.html#pyspark.SparkConf.contains)(key) | Does this configuration contain a given key? | | [**get**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.get.html#pyspark.SparkConf.get)(key[, defaultValue]) | Get the configured value for some key, or return a default otherwise. | | [**getAll**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.getAll.html#pyspark.SparkConf.getAll)() | Get all values as a list of key-value pairs. | | [**set**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.set.html#pyspark.SparkConf.set)(key, value) | Set a configuration property. | | [**setAll**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.setAll.html#pyspark.SparkConf.setAll)(pairs) | Set multiple parameters, passed as a list of key-value pairs. | | [**setAppName**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.setAppName.html#pyspark.SparkConf.setAppName)(value) | Set application name. | | [**setExecutorEnv**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.setExecutorEnv.html#pyspark.SparkConf.setExecutorEnv)([key, value, pairs]) | Set an environment variable to be passed to executors. | | [**setIfMissing**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.setIfMissing.html#pyspark.SparkConf.setIfMissing)(key, value) | Set a configuration property, if not already set. | | [**setMaster**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.setMaster.html#pyspark.SparkConf.setMaster)(value) | Set master URL to connect to. | | [**setSparkHome**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.setSparkHome.html#pyspark.SparkConf.setSparkHome)(value) | Set path where Spark is installed on worker nodes. | | [**toDebugString**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.SparkConf.toDebugString.html#pyspark.SparkConf.toDebugString)() | Returns a printable version of the configuration, as a list of key=value pairs, one per line. | |
| **Examples:**  **from** **pyspark.conf** **import** SparkConf  **>>> from** **pyspark.context** **import** SparkContext  **>>>** conf = SparkConf()  **>>>** conf.setMaster("local").setAppName("My app")  <pyspark.conf.SparkConf object at ...>  **>>>** conf.get("spark.master")  'local'  **>>>** conf.get("spark.app.name")  'My app'  **>>>** sc = SparkContext(conf=conf)  **>>>** sc.master  'local'  **>>>** sc.appName  'My app'  **>>>** sc.sparkHome **is** **None**  True  ---------------------------------------------------------------------------  **>>>** conf = SparkConf(loadDefaults=**False**) # to skip loading external settings  **>>>** conf.setSparkHome("/path")  <pyspark.conf.SparkConf object at ...>  **>>>** conf.get("spark.home")  '/path'  **>>>** conf.setExecutorEnv("VAR1", "value1")  <pyspark.conf.SparkConf object at ...>  **>>>** conf.setExecutorEnv(pairs = [("VAR3", "value3"), ("VAR4", "value4")])  <pyspark.conf.SparkConf object at ...>  **>>>** conf.get("spark.executorEnv.VAR1")  'value1'  **>>>** print(conf.toDebugString())  spark.executorEnv.VAR1=value1  spark.executorEnv.VAR3=value3  spark.executorEnv.VAR4=value4  spark.home=/path  **>>> for** p **in** sorted(conf.getAll(), key=**lambda** p: p[0]):  **...**  print(p)  ('spark.executorEnv.VAR1', 'value1')  ('spark.executorEnv.VAR3', 'value3')  ('spark.executorEnv.VAR4', 'value4')  ('spark.home', '/path')  **>>>** conf.\_jconf.setExecutorEnv("VAR5", "value5")  JavaObject id...  **>>>** print(conf.toDebugString())  spark.executorEnv.VAR1=value1  spark.executorEnv.VAR3=value3  spark.executorEnv.VAR4=value4  spark.executorEnv.VAR5=value5  spark.home=/path |

**pyspark.Broadcast**

A broadcast variable created with SparkContext.broadcast(). Access its value through value.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Methods**   |  |  | | --- | --- | | [**destroy**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Broadcast.destroy.html#pyspark.Broadcast.destroy)([blocking]) | Destroy all data and metadata related to this broadcast variable. | | [**dump**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Broadcast.dump.html#pyspark.Broadcast.dump)(value, f) |  | | [**load**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Broadcast.load.html#pyspark.Broadcast.load)(file) |  | | [**load\_from\_path**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Broadcast.load_from_path.html#pyspark.Broadcast.load_from_path)(path) |  | | [**unpersist**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Broadcast.unpersist.html#pyspark.Broadcast.unpersist)([blocking]) | Delete cached copies of this broadcast on the executors. |   **Attributes**   |  |  | | --- | --- | | [**value**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.Broadcast.value.html#pyspark.Broadcast.value) | Return the broadcasted value | |

|  |
| --- |
| **from** **pyspark.context** **import** SparkContext  **>>>** sc = SparkContext('local', 'test')  **>>>** b = sc.broadcast([1, 2, 3, 4, 5])  **>>>** b.value  [1, 2, 3, 4, 5]  **>>>** sc.parallelize([0, 0]).flatMap(**lambda** x: b.value).collect()  [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]  **>>>** b.unpersist()  >>> large\_broadcast = sc.broadcast(range(10000)) |

**pyspark.Accumulator**

**pyspark.AccumulatorParam**

**pyspark.SparkFiles**

**pyspark.StorageLevel**

**pyspark.RDDBarrier**

**pyspark.BarrierTaskContext**

**pyspark.BarrierTaskInfo**

**pyspark.InheritableThread**

**pyspark.util**