

1. 初始化 Spark

1.1 SparkContext

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



1.2 SparkContext 信息获取

```
sc.version          # 获取 SparkContext 版本
sc.pythonVer        # 获取 Python 版本
sc.master            # 要连接的 MasterURL
str(sc.sparkHome)    # Spark 在工作节点的安装路径
str(sc.sparkUser())  # 获取 SparkContext 的 Spark 用户名
```

```
sc.appName          # 返回应用名称
sc.applicationId     # 获取应用程序 ID
sc.defaultParallelism # 返回默认并行级别
sc.defaultMinPartitions # RDD 默认最小分区数
```

1.3 配置

```
from pyspark import SparkConf, SparkContext
conf = (SparkConf()
        .setMaster("local")
        .setAppName("My app")
        .set("spark.executor.memory", "1g"))
sc = SparkContext(conf=conf)
```

1.4 使用 Shell

PySpark Shell 已经为 SparkContext 创建了名为 sc 的变量。

命令行启动 spark

```
$ ./bin/spark-shell --master local[2]
```

命令行提交 spark 脚本任务

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

用 `--master` 参数设定 Context 连接到哪个 Master 服务器, 通过传递逗号分隔列表至 `--py-files` 添加 Python.zip、.egg 或 .py 文件到 Runtime 路径。

2. 加载数据

2.1 并行集合

```
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"]))
```

2.2 外部数据

使用 `textFile()` 函数从 HDFS、本地文件或其它支持 Hadoop 的文件系统里读取文本文件, 或使用 `wholeTextFiles()` 函数读取目录里文本文件。

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

3. 提取 RDD 信息

3.1 基础信息

```
rdd.getNumPartitions() # 列出分区数

rdd.count()             # 计算 RDD 实例数量
3

rdd.countByKey()        # 按键计算 RDD 实例数量
defaultdict(<type 'int'>, {'a':2, 'b':1})

rdd.countByValue()      # 按值计算 RDD 实例数量
defaultdict(<type 'int'>, {('b',2):1, ('a',2):1, ('a',7):1})

rdd.collectAsMap()      # 以字典形式返回键值
{'a': 2, 'b': 2}

rdd3.sum()              # RDD 元素求和
4950

sc.parallelize([]).isEmpty() # 检查 RDD 是否为空
True
```



扫码回复“数据科学”

下载最新全套速查表

3.2 汇总

```
rdd3.max()             # RDD 元素的最大值
99

rdd3.min()             # RDD 元素的最小值
0

rdd3.mean()            # RDD 元素的平均值
49.5

rdd3.stdev()           # RDD 元素的标准差
28.866070047722118

rdd3.variance()        # RDD 元素的方差
833.25

rdd3.histogram(3)      # 分箱 (Bin) 生成直方图
([0,33,66,99],[33,33,34])

rdd3.stats()           # 综合统计: 计数、标准差、平均 / 最大 / 最小值
```

4. 应用函数

map 与 flatmap 函数

对每个 RDD 元素执行函数

```
rdd.map(lambda x: x+(x[1], x[0])).collect()
[('a',7,7,'a'), ('a',2,2,'a'), ('b',2,2,'b')]
```

对每个 RDD 元素执行函数, 并拉平结果

```
rdd5=rdd.flatMap(lambda x: x+(x[1], x[0]))
rdd5.collect()
```

```
[('a',7,7,'a','a',2,2,'a','b',2,2,'b')]
```

不改变键, 对 rdd4 每个键值对执行 flatMap 函数

```
rdd4.flatMapValues(lambda x: x).collect()
[('a','x'), ('a','y'), ('a','z'), ('b','p'), ('b','r')]
```

Spark RDD 速查表

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

参考 | DataCamp Cheatsheet



5. 选择数据

5.1 获取

```
rdd.collect() # 返回包含所有 RDD 元素的列表
[('a', 7), ('a', 2), ('b', 2)]

# 提取前两个 RDD 元素
rdd.filter(lambda x: "a" in x).collect()
[('a', 7), ('a', 2)]

rdd.first() # 提取第一个 RDD 元素
('a', 7)

rdd5.distinct().collect() # 提取前两个 RDD 元素
['a', 2, 'b', 7]
```

5.2 抽样

```
# 返回 rdd3 的采样子集
rdd3.sample(False, 0.15, 81).collect()
[3, 4, 27, 31, 40, 41, 42, 43, 60, 76, 79, 80, 86, 97]
```

5.3 筛选

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

rdd5.distinct().collect() # 返回 RDD 里的唯一值
['a', 2, 'b', 7]

rdd.keys().collect() # 返回 RDD 键值对里的键
['a', 'a', 'b']
```

6. 迭代

```
foreach 函数迭代
def g(x):
    print(x)
rdd.foreach(g) # 为所有 RDD 应用函数
('a', 7)
('b', 2)
('a', 2)
```

7. 改变数据形状

7.1 Reduce 操作

```
rdd.reduceByKey(lambda x, y: x + y).collect() # 合并每个键的 RDD 值
[('a', 9), ('b', 2)]

rdd.reduce(lambda a, b: a + b) # 合并 RDD 的值
('a', 7, 'a', 2, 'b', 2)
```

7.2 分组

```
rdd3.groupBy(lambda x: x%2).mapValues(list).collect() # 返回 RDD 分组值
rdd.groupByKey().mapValues(list).collect() # 按键分组 RDD
[('a', [7, 2]), ('b', [2])]
```

7.3 聚合

```
seqOp = (lambda x, y: (x[0]+y, x[1]+1))
combOp = (lambda x, y: (x[0]+y[0], x[1]+y[1]))
add = (lambda x, y: x+y)
rdd3.aggregate((0, 0), seqOp, combOp) # 汇总每个分区的 RDD 元素，并输出结果
(4950, 100)

rdd.aggregateByKey((0, 0), seqOp, combOp).collect() # 汇总每个 RDD 键的值
[('a', (9, 2)), ('b', (2, 1))]

rdd3.fold(0, add) # 汇总每个分区里的 RDD 元素，并输出结果
4950

rdd.foldByKey(0, add).collect() # 合并每个键的值
[('a', 9), ('b', 2)]

rdd3.keyBy(lambda x: x+x).collect() # 通过执行函数，创建 RDD 元素的元组
```

8. 数学运算

RDD 运算

```
rdd.subtract(rdd2).collect() # 返回在 rdd2 里未匹配键的 rdd 键值对
[('b', 2), ('a', 7)]

rdd2.subtractByKey(rdd).collect() # 返回 rdd2 的每个键值对，rdd 未匹配的键
[('d', 1)]

rdd.cartesian(rdd2).collect() # 返回 rdd 和 rdd2 的笛卡尔积
```

9. 排序

RDD 排序

```
# 按给定函数排序
rdd2.sortBy(lambda x: x[1]).collect()
[('d', 1), ('b', 1), ('a', 2)]

# RDD 按键排序 RDD 的键值对
rdd2.sortByKey().collect()
[('a', 2), ('b', 1), ('d', 1)]
```

10. 重分区

repartition 函数

```
rdd.repartition(4) # 新建一个含 4 个分区的 RDD
rdd.coalesce(1) # 将 RDD 中的分区数缩减为 1 个
```

11. 保存

存储 RDD 到本地或 HDFS

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

12. 终止 SparkContext

停止 SparkContext

```
sc.stop()
```

13. 执行脚本程序

提交脚本执行

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

数据科学工具库速查表



NumPy 是 Python 数据科学计算的核心库，提供了高性能多维数组对象及处理数组的工具。使用以下语句导入 NumPy 库：

```
import numpy as np
```



SciPy 是基于 NumPy 创建的 Python 科学计算核心库，提供了众多数学算法与函数。



Pandas 是基于 NumPy 创建的 Python 库，为 Python 提供了易于使用的的数据结构和数据分析工具。使用以下语句导入：

```
import pandas as pd
```



Matplotlib 是 Python 的二维绘图库，用于生成符合出版质量或跨平台交互环境的各类图形。

```
import matplotlib.pyplot as plt
```



Seaborn 是基于 matplotlib 开发的高阶 Python 数据可视图库，用于绘制优雅、美观的统计图形。使用下列别名导入该库：

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```



Bokeh 是 Python 的交互式可视图库，用于生成在浏览器里显示的大规模数据集高性能可视图。Bokeh 的中间层通用 **bokeh.plotting** 界面主要为两个组件：数据与图示例。

```
from bokeh.plotting import figure
```

```
from bokeh.io import output_file, show
```



PySpark 是 Spark 的 Python API，允许 Python 调用 Spark 编程模型。Spark SQL 是 Apache Spark 处理结构化数据模块。

AI 垂直领域工具库速查表



Scikit-learn 是开源的 Python 库，通过统一的界面实现机器学习、预处理、交叉验证及可视化算法。



Keras 是强大、易用的深度学习库，基于 Theano 和 TensorFlow 提供了高阶神经网络 API，用于开发和评估深度学习模型。



“TensorFlow™ is an open source software library for numerical computation using data flow graphs.” **TensorFlow** 是 Google 公司开发的机器学习架构，兼顾灵活性和扩展性，既适合用于工业生产也适合用于科学研究。



PyTorch 是 Facebook 团队 2017 年初发布的深度学习框架，有利于研究人员、爱好者、小规模项目等快速搞出原型。**PyTorch** 也是 Python 程序员最容易上手的深度学习框架。



Hugging Face 以开源的 NLP 预训练模型库 **Transformers** 而广为人知，目前 GitHub Star 已超过 54000+。**Transformers** 提供 100+ 种语言的 32 种预训练语言模型，简单，强大，高性能，是新手入门的不二选择。



OpenCV 是一个跨平台计算机视觉库，由 C 函数 /C++ 类构成，提供了 Python、MATLAB 等语言的接口。**OpenCV** 实现了图像处理和计算机视觉领域的很多通用算法。

编程语言速查表



SQL 是管理关系数据库的结构化查询语言，包括数据的增删查改等。作为数据分析的必备技能、岗位 JD 的重要关键词，SQL 是技术及相关岗位同学一定要掌握的语言。



Python 编程语言简洁快速、入门简单且功能强大，拥有丰富的第三方库，已经成为大数据和人工智能领域的主流编程语言。

More...

AI 知识技能速查表



Jupyter Notebook 交互式计算环境，支持运行 40+ 种编程语言，可以用来编写漂亮的交互式文档。这个教程把常用的基础功能讲解得很清楚，对新手非常友好。



正则表达式 非常强大，能匹配很多规则的文本，常用于文本提取和爬虫处理。这也是一门令人难以捉摸的语言，字母、数字和符号堆在一起，像极了“火星文”。

More...



ShowMeAI 速查表 (©2021)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

数据科学工具库速查表

扫码回复“数据科学”

获取最新全套速查表

AI 垂直领域工具库速查表

扫码回复“工具库”

获取最新全套速查表

编程语言速查表

扫码回复“编程语言”

获取最新全套速查表

AI 知识技能速查表

扫码回复“知识技能”

获取最新全套速查表