基本RDD「轉換」運算

Step2 建立intRDD

```
val intRDD = sc.parallelize(List(3,1, 2, 5, 5))
intRDD.collect()
```

Step3 建立stringRDD

```
val stringRDD = sc.parallelize(List("Apple", "Orange",
"Banana","Grape","Apple"))
stringRDD.collect()
```

Step4 map運算

具名函數的寫法:

```
def addOne(x :Int):Int={
return (x+1)
}
intRDD.map(addOne).collect()
```

匿名函數的寫法:

```
intRDD.map(x => x + 1).collect()
```

匿名函數+匿名參數的寫法:

```
intRDD.map(_ + 1).collect()
```

Step7 filter字串運算

```
stringRDD.filter(x =>x.contains("ra") ).collect()
```

Step9 randomSplit運算

```
val sRDD = intRDD.randomSplit(Array(0.4,0.6))
sRDD(0).collect()
sRDD(1).collect()
```

Step10 groupBy運算

```
val gRDD=intRDD.groupBy(
x =>{ if (x % 2 == 0) "even" else "odd"}
).collect
gRDD(0)
gRDD(1)
```

多個RDD「轉換」運算

Step1 建立3個範例RDD

```
val intRDD1 = sc.parallelize(List(3, 1, 2, 5, 5))
val intRDD2 = sc.parallelize(List(5, 6))
val intRDD3 = sc.parallelize(List(2, 7))
```

Step2 union 聯集運算

```
intRDD1.union(intRDD2).union(intRDD3).collect()
(intRDD1 ++ intRDD2++ intRDD3).collect()
```

Step3 intersection 交集運算

```
intRDD1.intersection(intRDD2).collect()
```

Step4 subtract 差集運算

```
intRDD1.subtract(intRDD2).collect()
```

Step5 cartesian笛卡兒乘積運算

```
intRDD1.cartesian(intRDD2).collect()
```

基本「動作」運算

Step1 讀取元素

```
intRDD.first
intRDD.take(2)
intRDD.takeOrdered(3)
intRDD.takeOrdered(3)(Ordering[Int].reverse)
```

Step2 統計功能

```
intRDD.stats
intRDD.min
intRDD.max
intRDD.stdev
intRDD.count
intRDD.sum
intRDD.mean
```

RDD Key-Value 基本「轉換」運算

Step1 讀取元素
```
val kvRDD1 = sc.parallelize(List((3, 4), (3, 6), (5, 6), (1, 2)))
```

Step2 列出keys值
```
kvRDD1.keys.collect()
```

Step3 列出values值
```
kvRDD1.values.collect()
```

Step4 使用filter篩選key運算
```
kvRDD1.filter { case (key, value) => key < 5 }.collect
```

Step5 使用filter篩選value運算
```
kvRDD1.filter { case (key, value) => value < 5 }.collect
```

Step6 mapValues運算
```
kvRDD1.mapValues(x => x * x).collect
```

Step7 sortByKey由小至大依照key排序
```
kvRDD1.sortByKey(true).collect()
kvRDD1.sortByKey().collect()
```

Step8 sortByKey由大至小依照key排序
```
kvRDD1.sortByKey(false).collect()
```

Step9 reduceByKey
```
kvRDD1.reduceByKey((x,y)=>x+y).collect
kvRDD1.reduceByKey(_+_).collect
```

多個RDD Key-Value「轉換」運算

Step1 Key-Value RDD範例
```
val kvRDD1 = sc.parallelize(List((3, 4), (3, 6), (5, 6), (1, 2)))
val kvRDD2 = sc.parallelize(List((3, 8)))
```

Step2 Key-Value RDD join運算
kvRDD1.join(kvRDD2).foreach(println)

Step2 Key-Value RDD join運算
kvRDD1.join(kvRDD2).foreach(println)

Step3 Key-Value leftOuterJoin運算
kvRDD1.leftOuterJoin(kvRDD2).foreach(println)

Step4 Key-Value RDD rightOuterJoin運算
kvRDD1.rightOuterJoin(kvRDD2).foreach(println)

Step5 Key-Value subtractByKey運算
kvRDD1.subtractByKey(kvRDD2).collect

Key-Value 「動作」運算

Step1 Key-Value first運算
kvRDD1.first()
kvRDD1.take(2)

Step2 取得第1筆資料的元素
val kvFirst=kvRDD1.first
kvFirst._1
kvFirst._2

Step3 Key-Value countByKey運算
kvRDD1.countByKey()

Step4 Key-Value collectAsMap運算
var KV=kvRDD1.collectAsMap()

Step5 使用對照表轉換資料
KV(3)
KV(4)

Step6 Key-Value lookup運算

```
kvRDD1.lookup(3)
kvRDD1.lookup(5)
```