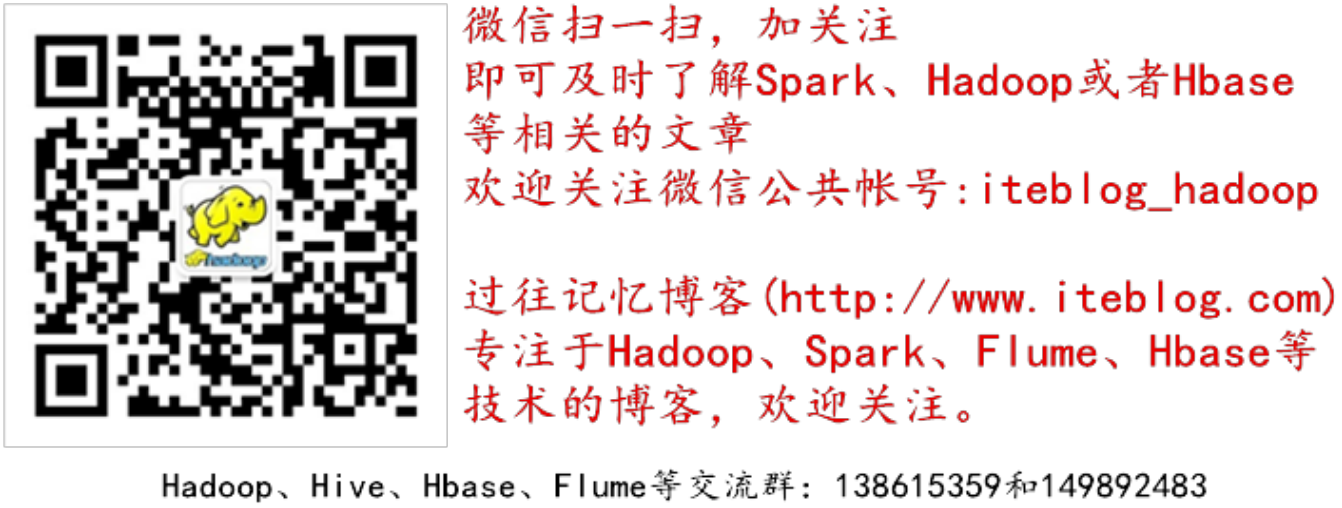
Spark Python API函数学习：pyspark API(1)

Spark支持Scala、Java以及Python语言，本文将通过图片和简单例子来学习pyspark API。



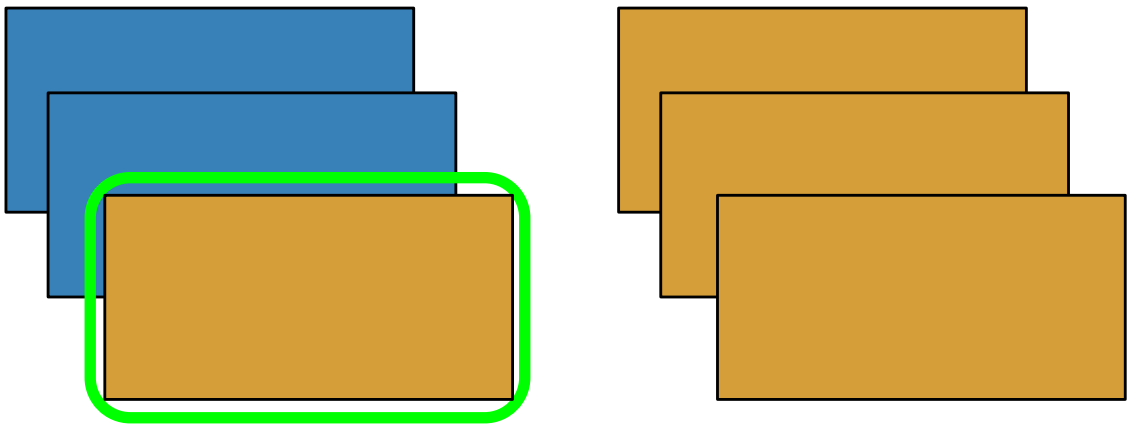
如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

pyspark version

```
# print Spark version
print("pyspark version:" + str(sc.version))
```

pyspark version:1.2.2

map

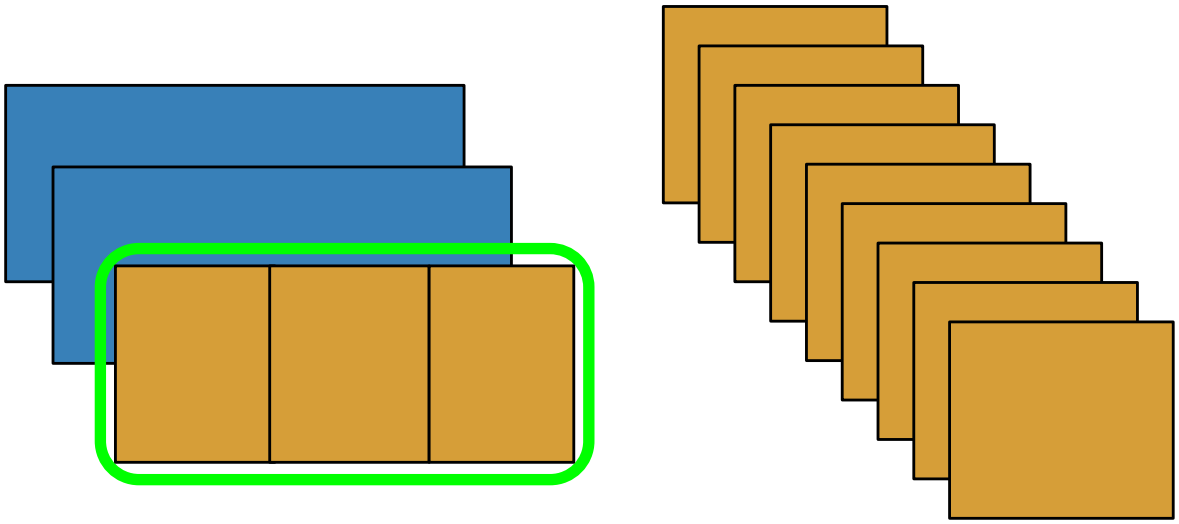


```
# map
# sc = spark context, parallelize creates an RDD from the passed object
x = sc.parallelize([1,2,3])
y = x.map(lambda x: (x,x**2))

# collect copies RDD elements to a list on the driver
print(x.collect())
print(y.collect())
```

```
[1, 2, 3]
[(1, 1), (2, 4), (3, 9)]
```

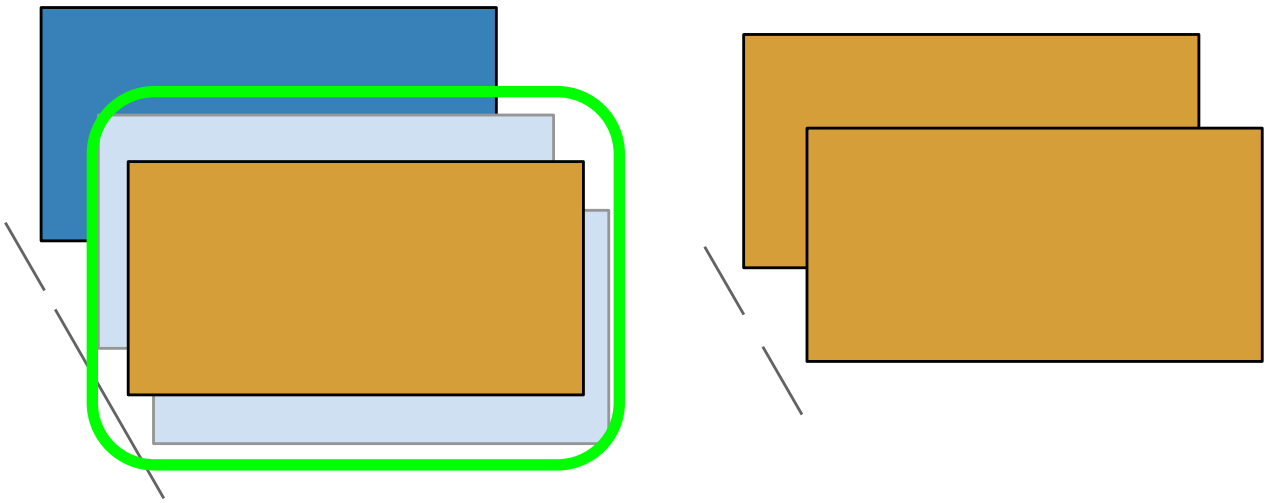
flatMap



```
# flatMap
x = sc.parallelize([1,2,3])
y = x.flatMap(lambda x: (x, 100*x, x**2))
print(x.collect())
print(y.collect())

[1, 2, 3]
[1, 100, 1, 2, 200, 4, 3, 300, 9]
```

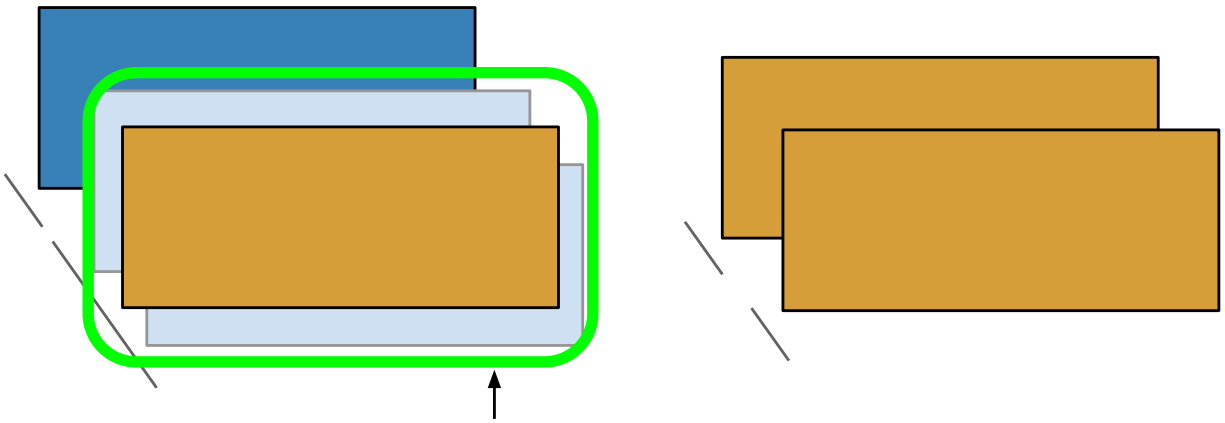
mapPartitions



```
# mapPartitions
x = sc.parallelize([1,2,3], 2)
def f(iterator): yield sum(iterator)
y = x.mapPartitions(f)
# glom() flattens elements on the same partition
print(x.glom().collect())
print(y.glom().collect())

[[1], [2, 3]]
[[1], [5]]
```

mapPartitionsWithIndex

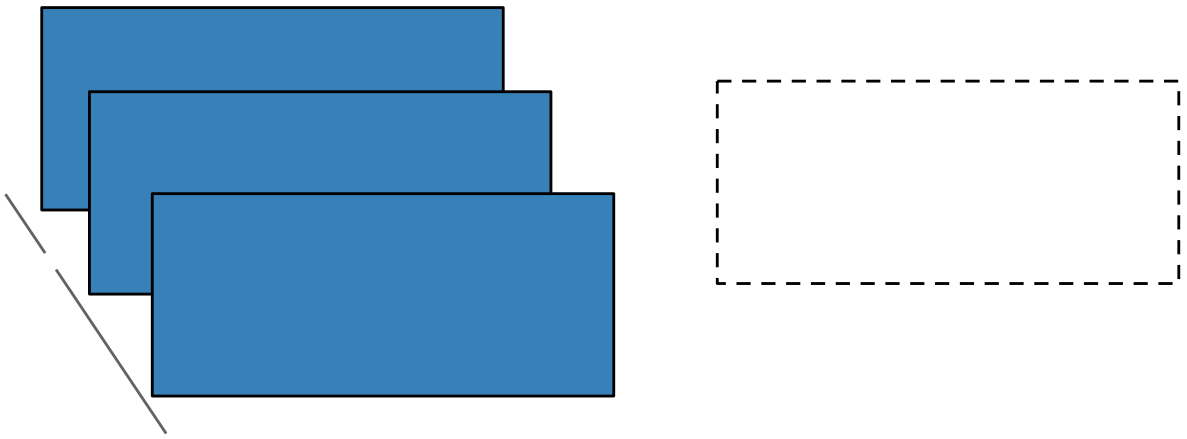


```
# mapPartitionsWithIndex
x = sc.parallelize([1,2,3], 2)
def f(partitionIndex, iterator): yield (partitionIndex,sum(iterator))
y = x.mapPartitionsWithIndex(f)

# glom() flattens elements on the same partition
print(x.glom().collect())
print(y.glom().collect())

[[1], [2, 3]]
[[(0, 1)], [(1, 5)]]
```

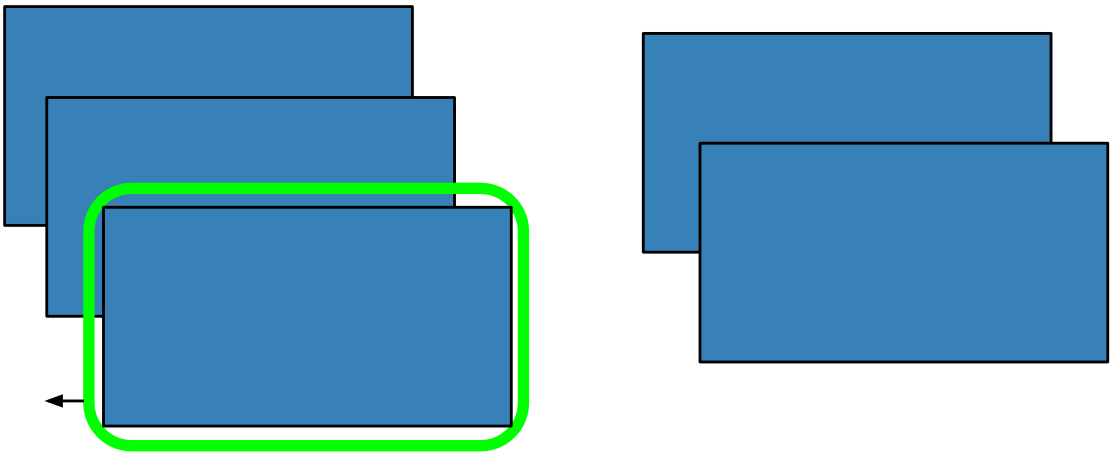
getNumPartitions



```
# getNumPartitions
x = sc.parallelize([1,2,3], 2)
y = x.getNumPartitions()
print(x.glom().collect())
print(y)

[[1], [2, 3]]
2
```

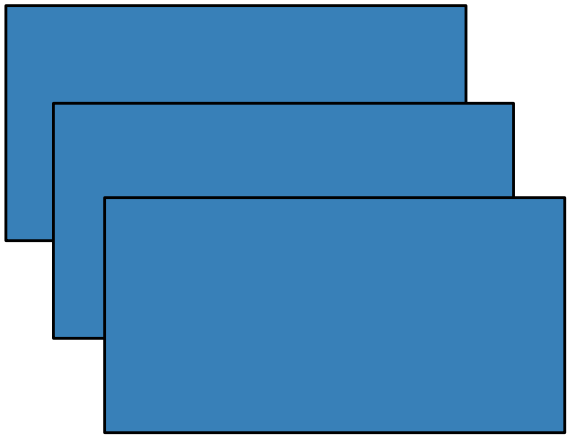
filter



```
# filter
x = sc.parallelize([1,2,3])
y = x.filter(lambda x: x%2 == 1)  # filters out even elements
print(x.collect())
print(y.collect())

[1, 2, 3]
[1, 3]
```

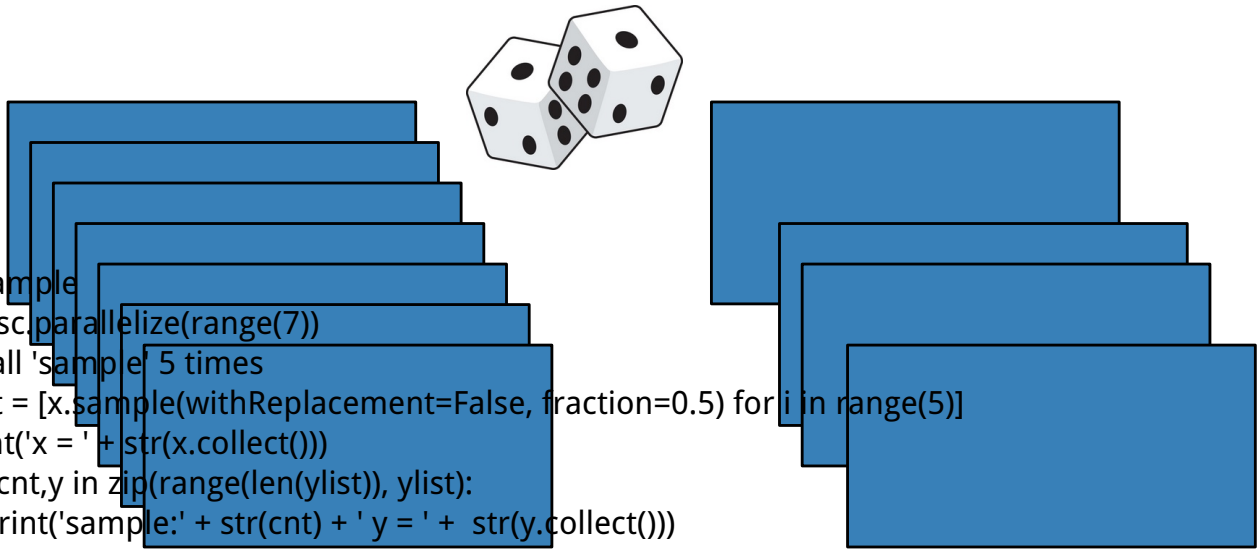
distinct



```
# distinct
x = sc.parallelize(['A','A','B'])
y = x.distinct()
print(x.collect())
print(y.collect())

['A', 'A', 'B']
['A', 'B']
```

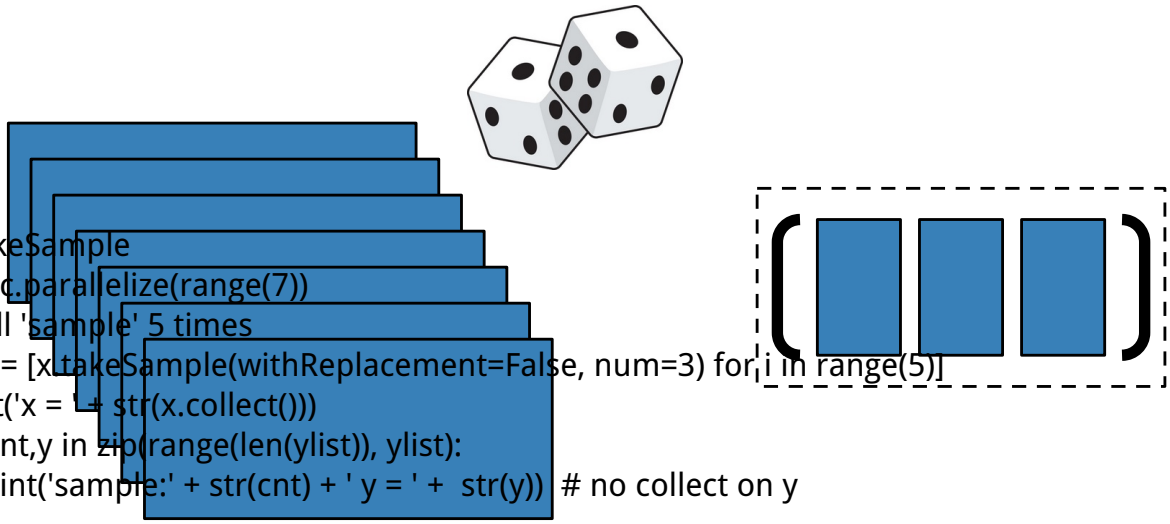
sample



```
# sample
x = sc.parallelize(range(7))
# call 'sample' 5 times
ylist = [x.sample(withReplacement=False, fraction=0.5) for i in range(5)]
print('x = ' + str(x.collect()))
for cnt,y in zip(range(len(ylist)), ylist):
    print('sample:' + str(cnt) + ' y = ' +  str(y.collect()))
```

```
x = [0, 1, 2, 3, 4, 5, 6]
sample:0 y = [0, 2, 5, 6]
sample:1 y = [2, 6]
sample:2 y = [0, 4, 5, 6]
sample:3 y = [0, 2, 6]
sample:4 y = [0, 3, 4]
```

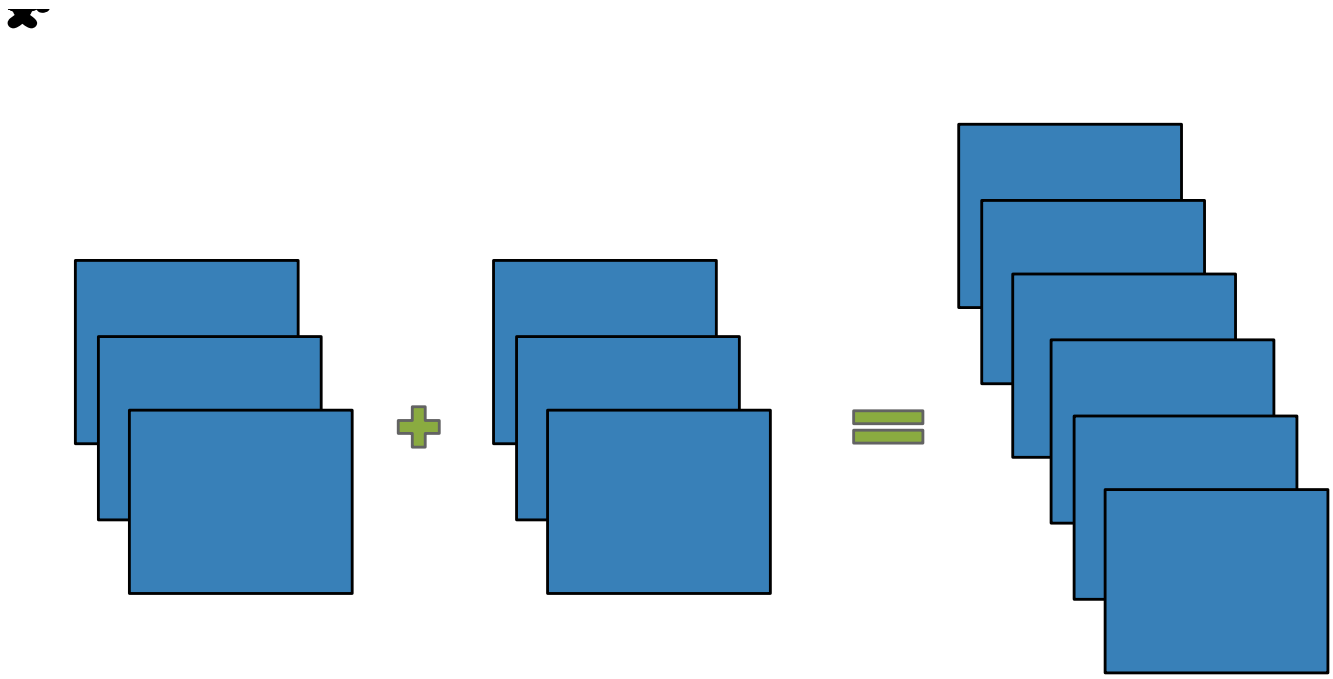
takeSample



```
# takeSample
x = sc.parallelize(range(7))
# call 'sample' 5 times
ylist = [x.takeSample(withReplacement=False, num=3) for i in range(5)]
print('x = ' + str(x.collect()))
for cnt,y in zip(range(len(ylist)), ylist):
    print('sample:' + str(cnt) + ' y = ' +  str(y))  # no collect on y
```

```
x = [0, 1, 2, 3, 4, 5, 6]
sample:0 y = [0, 2, 6]
sample:1 y = [6, 4, 2]
sample:2 y = [2, 0, 4]
sample:3 y = [5, 4, 1]
sample:4 y = [3, 1, 4]
```

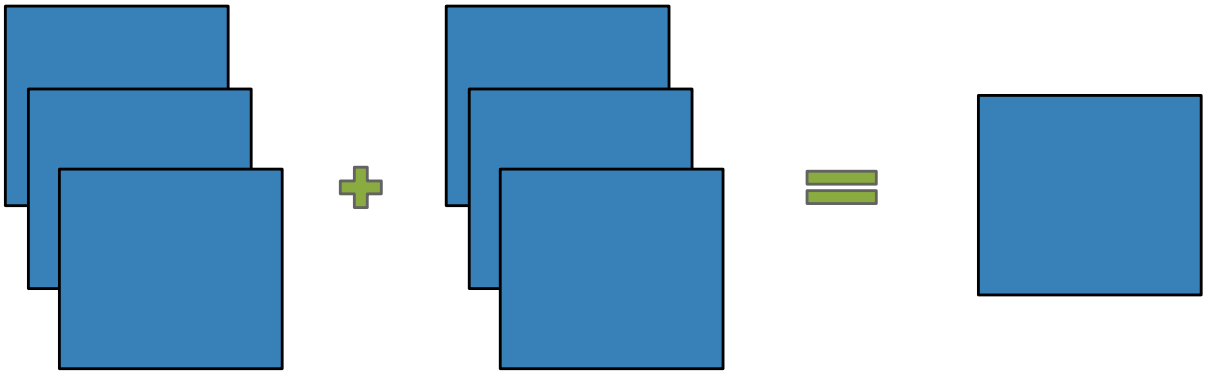
union



```
# union
x = sc.parallelize(['A','A','B'])
y = sc.parallelize(['D','C','A'])
z = x.union(y)
print(x.collect())
print(y.collect())
print(z.collect())

['A', 'A', 'B']
['D', 'C', 'A']
['A', 'A', 'B', 'D', 'C', 'A']
```

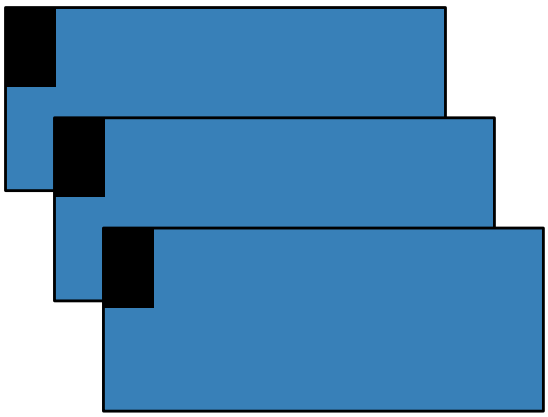
intersection

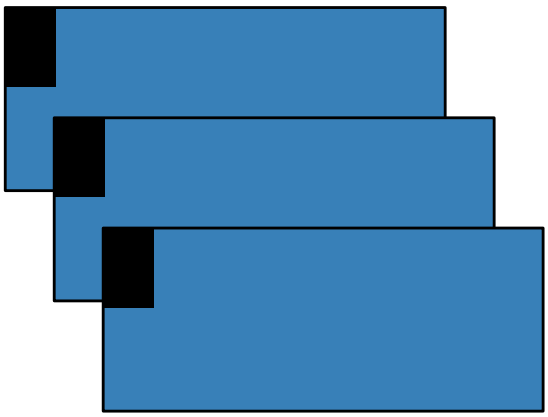


```
# intersection
x = sc.parallelize(['A','A','B'])
y = sc.parallelize(['A','C','D'])
z = x.intersection(y)
print(x.collect())
print(y.collect())
print(z.collect())

['A', 'A', 'B']
['A', 'C', 'D']
['A']
```

sortByKey





```
# sortByKey
x = sc.parallelize([('B',1),('A',2),('C',3)])
y = x.sortByKey()
print(x.collect())
print(y.collect())

[('B', 1), ('A', 2), ('C', 3)]
[('A', 2), ('B', 1), ('C', 3)]
```