02.RDD

segunda-feira, 25 de setembro de 2023 10:21

```
>>> numeros = sc.parallelize([1,2,3,4,5,6,7,8,9,10])
>>> numeros.take(5)
[1, 2, 3, 4, 5]
>>> numeros.top(5)
[10, 9, 8, 7, 6]
>>> numeros.collect()
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> numeros.count()
10
>>> numeros.mean()
5.5
>>> numeros.sum()
55
>>> numeros.max()
10
>>> numeros.min()
1
>>> numeros.stdev()
2.8722813232690143
```

Sc.parallelize - Cria um RDD
.take - Pega os n primeiros argumentos
.top - Pega os n ultimos argumentos
.collect() - pega todos os argumentos
.**count()** - conta os argumentos
.**mean()** - Media dos numeros
.sum() - Soma os numeros
.max() - Pega o maior valor
.min() - Pega o menor valor

```
>>> filtro = numeros.filter(lambda filtro: filtro > 2 )
>>> filtro.collect()
[3, 4, 5, 6, 7, 8, 9, 10]
>>> amostra = numeros.sample(True,0.5,1)
>>> amostra.collect()
[2, 3, 4, 5, 9, 10]
>>>
```

Usando filtros

```
>>> mapa = numeros.map(lambda mapa: mapa * 2)
>>> mapa.collect()
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
>>>
```

```
>>> numeros2 = sc.parallelize([6,7,8,9,10])
>>> uniao = numeros.union(numeros2)
>>> uniao.collect()
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 6, 7, 8, 9, 10]
>>> interseccao = numeros.intersection(numeros2)
>>> interseccao.collect()
[6, 7, 8, 9, 10]
>>> subtrai = numeros.subtract(numeros2)
>>> subtrai.collect()
[1, 2, 3, 4, 5]
>>> cartesiano = numeros.cartesian(numeros2)
>>> cartesiano.collect()
[(1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (2, 6), (2, 7), (2, 8), (2, 9), (2, 10), (3, 6), (3, 7), (3, 8), (3,
0), (4, 6), (5, 6), (4, 7), (5, 7), (4, 8), (5, 8), (4, 9), (5, 9), (4, 10), (5, 10), (6, 6), (6, 7), (6, 8),
, 10), (7, 6), (7, 7), (7, 8), (7, 9), (7, 10), (8, 6), (8, 7), (8, 8), (8, 9), (8, 10), (9, 6), (10, 6), (9,
), (9, 8), (10, 8), (9, 9), (10, 9), (9, 10), (10, 10)]
>>> cartesiano.countByValue()
defaultdict(<class 'int'>, {(1, 6): 1, (1, 7): 1, (1, 8): 1, (1, 9): 1, (1, 10): 1, (2, 6): 1, (2, 7): 1, (2,
 9): 1, (2, 10): 1, (3, 6): 1, (3, 7): 1, (3, 8): 1, (3, 9): 1, (3, 10): 1, (4, 6): 1, (5, 6): 1, (4, 7): 1, (
(4, 8): 1, (5, 8): 1, (4, 9): 1, (5, 9): 1, (4, 10): 1, (5, 10): 1, (6, 6): 1, (6, 7): 1, (6, 8): 1, (6, 9): 1
 1, (7, 6): 1, (7, 7): 1, (7, 8): 1, (7, 9): 1, (7, 10): 1, (8, 6): 1, (8, 7): 1, (8, 8): 1, (8, 9): 1, (8, 10
6): 1, (10, 6): 1, (9, 7): 1, (10, 7): 1, (9, 8): 1, (10, 8): 1, (9, 9): 1, (10, 9): 1, (9, 10): 1, (10, 10):
>>>
```

Lista1.union(lista2) - Unindo 2 listas
Lista1.intersection(lista2) - Verificando números em comum nas 2 listas
Lista1.cartesian(lista2) - verificando numeros cartesianos
Lista1.subtract(lista2) - subtraindo entre listas
Cartesiano.countByValue() - contando elementos e quantas vezes se repetem

```
>>> compras = sc.parallelize([(1,200),(2,300),(3,120),(4,250),(5,78)])
>>> chaves = compras.keys()
>>> chaves.collect()
[1, 2, 3, 4, 5]
>>> valores = compras.values()
>>> valores.collect()
[200, 300, 120, 250, 78]
>>> compras.countByKey()
defaultdict(<class 'int'>, {1: 1, 2: 1, 3: 1, 4: 1, 5: 1})
>>> compras.countByValues()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'RDD' object has no attribute 'countByValues'. Did you mean: 'countByValue'?
>>> compras.countByValue()
defaultdict(<class 'int'>, {(1, 200): 1, (2, 300): 1, (3, 120): 1, (4, 250): 1, (5, 78): 1})
>>> soma = compras.mapValues(lambda soma: soma + 1)
>>> soma.collect()
[(1, 201), (2, 301), (3, 121), (4, 251), (5, 79)]
>>> compras.collect()
[(1, 200), (2, 300), (3, 120), (4, 250), (5, 78)]
>>> debitos = sc.parallelize([(1,20),(2,300)])
>>> resultado = compras.join(debitos)
>>> resultado.collect()
[(1, (200, 20)), (2, (300, 300))]
>>> semdebito = compras.subtractByKey(debitos)
>>> semdebito.collect()
[(3, 120), (4, 250), (5, 78)]
>>>
```

.keys() - Pegando apenas chaves
.values() - Pegando apenas valores
.countByKey() - Contando chaves
.mapValues() - criando map para somar apenas aos valores
.join() - Juntando duas RDD
.