# Spark RDD(16/11/2018)

## Carga de datos

```
In []: lista=[3,6,2,7,9]
       listaRDD=sc.parallelize(lista)
```
Conjunto de datos

```
In []: ventas=[("moto",2000),
               ("coche",10000),
               ("coche",12000),
               ("moto",3000),
               ("moto",2000),
                ("bici",200)
               ]
       ventasRDD=sc.parallelize(ventas)
```
Similar a tabla. RDD de tuplas donde cada tupla representa una fila y cada elemento de la tupla se considera una columna en función de la posición.

## Datos externos

```
In []: lineas=sc.textFile("usuarios.csv")
```
RDD de líneas del archivo

```
In []: archivos=sc.wholetextFiles("/carpeta")
```
RDD de pares nombre y contenido

## Recuperando información

```
In []: listaRDD.collect()
Out[]: [3,6,2,7,9]
```
Devuelve el conjunto como lista.

```
In []: ventasRDD.first()
Out[]: ("moto",2000)
```
Primer elemento del RDD

```
In []: listaRDD.take(3)
Out[]: [3,6,2]
```
3 primeros elementos como lista

```
In []: ventasRDD.count()
Out[]: 6
```
Número de elementos del RDD

```
In []: listaRDD.sum()
Out[]: 27
```
Suma los elementos del RDD

```
In []: listaRDD.max()
Out[]: 9
```
Máximo del RDD

```
In []: listaRDD.min()
Out[]: 2
```
Mínimo del RDD

```
In []: listaRDD.mean()
Out[]: 5.4
```
Media del RDD

```
In []: listaRDD.stdev()
Out[]: 2.5768197453450252
```
Máximo del RDD

```
In []: listaRDD.stats()
Out[]: (count: 5, mean: 5.4, stdev:
       2.5768197453450252, max: 9.0, min: 2.0)
```
Conjunto de valores estadísticos

```
In []: listaRDD.variance()
Out[]: 6.640000000000001
```
Varianza del RDD

## Transformaciones del RDD

```
In []: dobleRDD=listaRDD.map(lambda a:a*2)
       dobleRDD.collect()
Out[]: [6,12,4,14,18]
```
Devuelve un RDD transformado según la función lambda

```
In []: filtroRDD=listaRDD.filter(lambda a:a>2)
       filtroRDD.collect()
Out[]: [3,6,7,9]
```
Devuelve un RDD filtrado según la condición de la función lambda

```
In []: txt=sc.parallelize(["Hola a todos"])
In []: palabras=txt.flatMap(lambda a:a.split(" "))
       palabras.collect()
Out[]: ["Hola","a","todos"]
```
Devuelve un RDD en el que cada elemento de la lista forma parte del nuevo RDD

## Claves y valores

```
In []: claves = ventasRDD.keys()
       claves.collect()
Out[]: ["moto","coche","coche","moto","moto","bici"]
```
Devuelve la primera columna (claves) como RDD

```
In []: valores = ventasRDD.values()
       valores.collect()
Out[]: [2000,10000,12000,3000,2000,200]
```
Devuelve la segunda columna (valores) como RDD

```
In []: nColumna=ventasRDD.map(lambda a:a[1])
```
Devuelve la columna indicada como RDD (empezando en 0)

```
In []: listaRDD.reduce(lambda a,b:a+b)
Out[]: 27
```
Realiza la operación indicada en lambda para todos los elementos

```
In []: res1=ventasRDD.reduceByKey(lambda a,b:a+b)
       res1.collect()
Out[]: [('moto', 7000), ('coche', 22000), ('bici', 200)]
```
Agrupa por clave y realiza la operación indicada en lambda con los valores

```
In []: porClave=res1.sortByKey()
       porClave.collect()
Out[]: [('bici', 200), ('coche', 22000), ('moto', 7000)]
```
Ordena por clave

```
In []: porValor=res1.sortBy(lambda a:-a[1])
       porValor.collect()
Out[]: [('coche', 22000), ('moto', 7000), ('bici', 200)]
```
Ordena por valor, de mayor a menor

## Python. Librerías útiles con RDD

```
In []: from datetime import datetime
       fecha=datetime.strptime('2/15/1999',"%m/%d/%Y")
       fecha
Out[]: datetime.datetime(1999, 2, 15, 0, 0)
```
Convierte una cadena de texto a objeto fecha

```
In []: fecha.weekday()
Out[]: 0
```
Día de la semana(0=lunes,...)

```
In []: import json
       texto='{"Nombre":"Pedro","Edad":25,"Aficiones":["
       Cine","Lectura"]}'
       datos=json.loads(texto)
       datos.get("Nombre"),datos.get("Aficiones")[1]
Out[]: ('Pedro', 'Lectura')
```
Interpreta json y convierte a diccionario.
Con datos.items() se obtendría una lista de tuplas con los valores del diccionario.

```
In []: import re
       patronDNI=re.compile(r"(\d{8})([A-Z])")
       texto="El DNI de Felipe es 45678912W y \
       el de María es 56432135A"
       patronDNI.findall(texto)
Out[]: [('45678912', 'W'), ('56432135', 'A')]
```
Expresiones regulares para encontrar información