



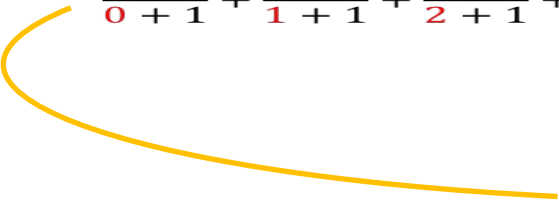
Programación distribuida con map-reduce

Contenido

- Introducción
- Las piezas básicas: map, reduce, yield, etiquetas
- Ejemplos: básico, ..., más realista, más completo
- Propiedades matemáticas básicas
- Estudio de tres casos resueltos

Introducción

$$\frac{0^2}{0+1} + \frac{1^2}{1+1} + \frac{2^2}{2+1} + \dots + \frac{(n-1)^2}{n-1+1}$$

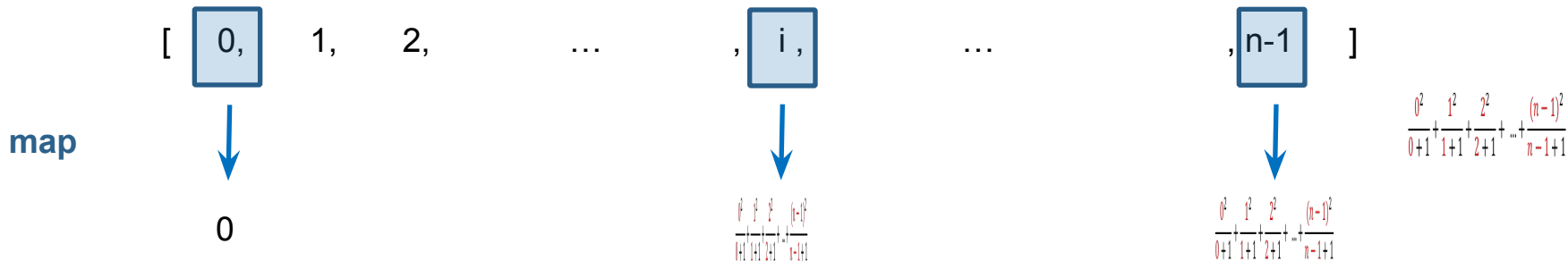

$$\frac{0^2}{0+1} + \frac{1^2}{1+1} + \frac{2^2}{2+1} + \dots + \frac{(n-1)^2}{n-1+1}$$

$$\frac{0^2}{0+1} + \frac{1^2}{1+1} + \frac{2^2}{2+1} + \dots + \frac{(n-1)^2}{n-1+1}$$

```
34 def sumatorio(n):  
35     acum = 0  
36     for i in range(n):  
37         acum = acum + i**2/(i+1)  
38     return suma
```

```
41 def polinomio(n, x):  
42     acum = 0  
43     for i in range(1, n+1):  
44         acum = acum*x + i  
45     return suma
```

Las piezas básicas: map



```
def lista_de_terminos(n):  
    return [i**2/(i+1) for i in range(n)]
```

```
print(lista_de_terminos(5))
```

```
def termino(i):  
    return i**2/(i+1)
```

```
def lista_de_terminos(n):  
    return list(map(termino, range(n)))
```

```
print(lista_de_terminos(5))
```

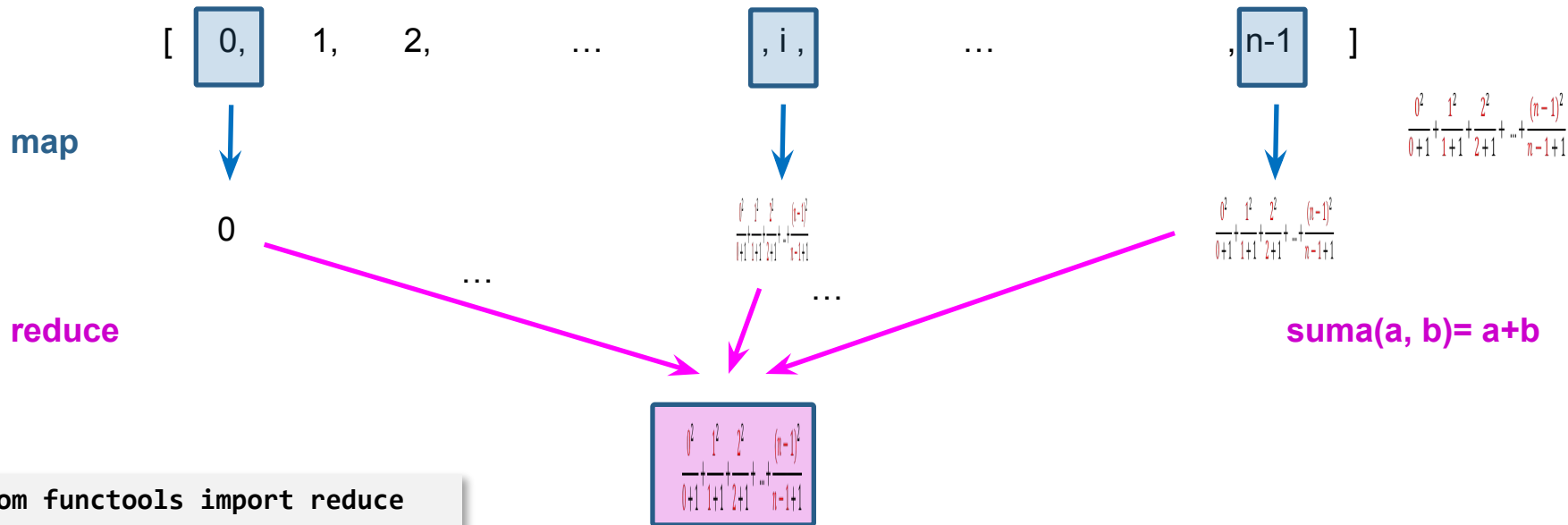
Con una lista intensional

[0.0, 0.5, 1.3333333333333333, 2.25, 3.2]

Con la función *map*

[0.0, 0.5, 1.3333333333333333, 2.25, 3.2]

Las piezas básicas: reduce



```
from functools import reduce
```

```
def suma(a, b):  
    return a+b
```

```
def sumatorio(lista):  
    return reduce(suma, lista)
```

```
mis_terminos = lista_de_terminos(5)  
print(lista_de_terminos(5))
```

```
mi_sumatorio = sumatorio(mis_terminos)  
print(mi_sumatorio)
```

$[0.0, 0.5, \dots, 2.25, 3.2]$

11.45

Las piezas básicas: yield

```
def natural_numbers():  
    i = 0  
    while True:  
        yield i  
        i = i+1  
  
nats = natural_numbers()
```

```
for _ in range(5):  
    print(next(nats))  
  
print(".....")  
  
for _ in range(5):  
    print(next(nats))
```

```
0  
1  
2  
3  
4  
.....  
5  
6  
7  
8  
9
```

nats = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...)

next(nats) 0

nats (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...)

next(nats) 1

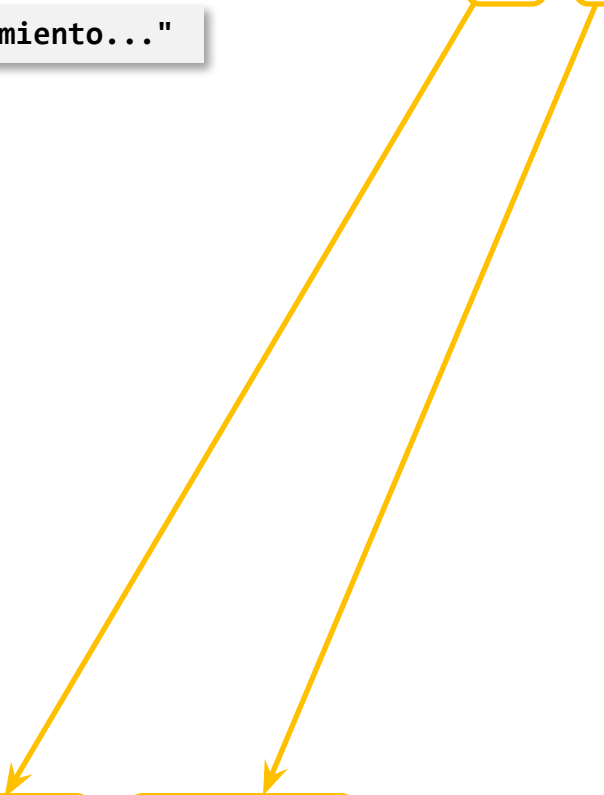
nats (2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...)

Las piezas básicas: etiquetas

```
frase = "Muchos años después, frente al pelotón de fusilamiento..."
```

[11 , 46]

[('otros', 11), ('letra', 46)]



Las piezas básicas: etiquetas

```
frase = "Muchos años después, frente al pelotón de fusilamiento..."
```

```
unos = list(map(lambda letra: 1, frase))  
print(unos)
```

```
total = reduce(suma, list(unos))  
print(total)
```

```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]  
57
```

[11 , 46]

```
frase = "Muchos años después, frente al pelotón de fusilamiento..."
```

```
def clase(caracter):  
    if caracter.isalpha():  
        return "letra", 1  
    else:  
        return "otros", 1
```

```
[('letra', 1), ('letra', 1), ('letra', 1), ('letra', 1), ('letra', 1),  
(('letra', 1), ('otros', 1), ('letra', 1), ('letra', 1), ('letra', 1),  
(('letra', 1), ('otros', 1), ('letra', 1), ('letra', 1), ('letra', 1),  
(('letra', 1), ('letra', 1), ('letra', 1), ...]
```

```
claves_unos = list(map(clase, frase))  
print(claves_unos)
```

[('otros', 11), ('letra', 46)]

Las piezas básicas: etiquetas

```
frase = "Muchos años después, frente al pelotón de fusilamiento..."
```

```
def clase(caracter):
    if caracter.isalpha():
        return "letra", 1
    else:
        return "otros", 1
```

```
claves_unos = list(map(clase, frase))
print(claves_unos)
```

```
def separar(lista_de_pares):
    claves = {k for k, v in lista_de_pares}
    return [(k, [v for (clave, v) in lista_de_pares if k ==clave]) for k in
claves]
```

```
lista_de_pares = separar(claves_unos)
print(separar(claves_unos))
```

```
[('otros', [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]),  
 ('letra', [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])]
```

```
def reduce_por_claves(lista_de_pares):
    return [(key, sum(valores)) for (key, valores) in lista_de_pares]
```

```
print(reduce_por_claves(lista_de_pares))
```

```
[('otros', 11), ('letra', 46)]
```

Un primer ejemplo completo

¿Cuántas letras y otros caracteres contiene un texto?

one_hundred_years.txt

```
one_hundred_years: Bloc de notas
Archivo  Editar  Ver
Many years later,
as he faced the firing squad,
Colonel Aureliano Buendia was to remember
that distant afternoon
when his father took him to discover ice.
```

char_classifier.py

```
from mrjob.job import MRJob
```

```
class MRWordCount(MRJob):
```

```
    def mapper(self, _, linea):
        for caracter in linea:
            if caracter.isalpha():
                yield "letra", 1
            else:
                yield "otros", 1
```

```
    def reducer(self, key, valores):
        yield key, sum(valores)
```

```
if __name__ == '__main__':
    MRWordCount.run()
```

```
'otros', (1, 1, 1, ...,
1)
'letra', (1, 1, 1, ...,
1)
```

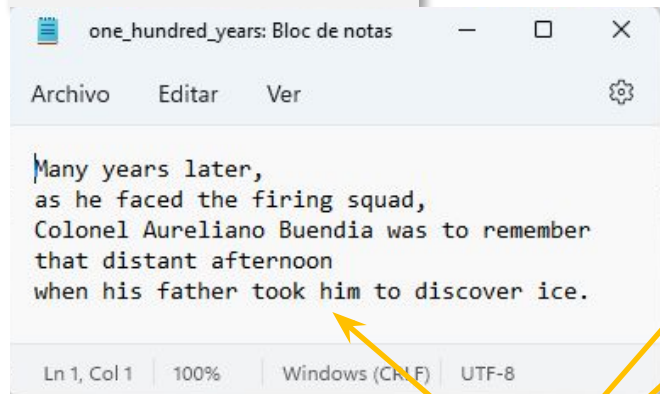
```
! python ".\char_classifier.py" ".\one_hundred_years.txt" -q
```

```
"letra" 126
"otros" 24
```

Un primer ejemplo completo

¿Cuántas letras y otros caracteres contiene un texto?

one_hundred_years.txt



char_classifier.py

```
from mrjob.job import MRJob  
  
class MRWordCount(MRJob):  
  
    def mapper(self, _, linea):  
        for caracter in linea:  
            if caracter.isalpha():  
                yield "letra", 1  
            else:  
                yield "otros", 1  
  
    def reducer(self, key, valores):  
        yield key, sum(valores)  
  
if __name__ == '__main__':  
    MRWordCount.run()
```

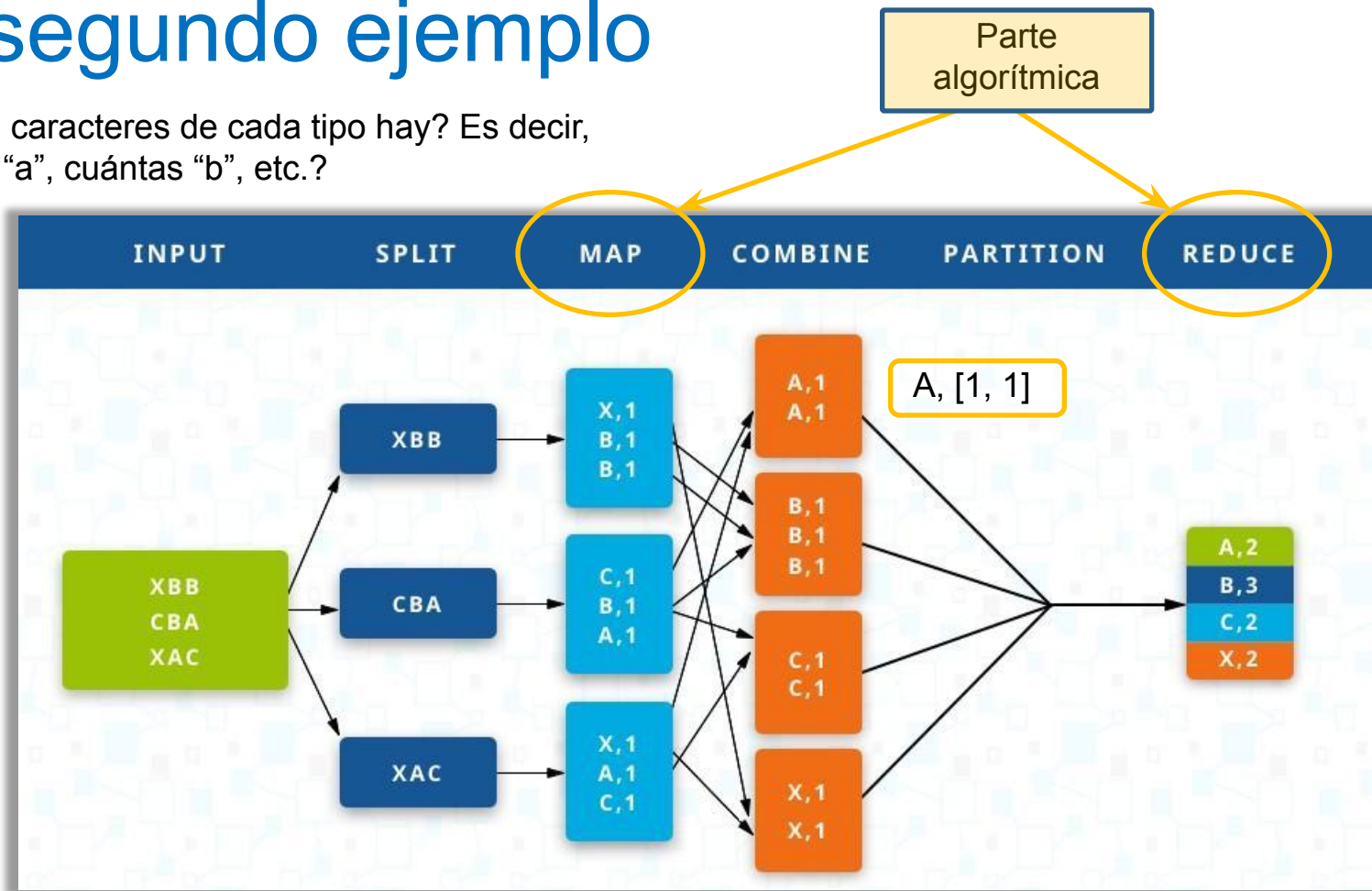
Observaciones

```
! python ".\char_classifier.py" ".\one_hundred_years.txt" -q
```

```
"letra" 126  
"otros" 24
```

Un segundo ejemplo

¿Cuántos caracteres de cada tipo hay? Es decir,
¿cuántas “a”, cuántas “b”, etc.?



Un segundo ejemplo

¿Cuántos caracteres de cada tipo hay? Es decir, ¿cuántas “a”, cuántas “b”, etc.?

one_hundred_years.txt



```
one_hundred_years: Bloc de notas
Archivo  Editar  Ver
Many years later,
as he faced the firing squad,
Colonel Aureliano Buendia was to remember
that distant afternoon
when his father took him to discover ice.
```

char_count.py

```
from mrjob.job import MRJob

class MRCharCount(MRJob):

    def mapper(self, _, linea):
        for caracter in linea:
            yield caracter, 1

    def reducer(self, key, valores):
        yield key, sum(valores)

if __name__ == '__main__':
    MRCharCount.run()
```

```
! python ".\char_count.py" ".\one_hundred_years.txt" -q
```

```
" " 21
", " 2
"." 1
"A" 1
"B" 1
```

```
"C" 1
"M" 1
"a" 13
"b" 1
"c" 3
```

```
"d" 5
"e" 16
"f" 4
"g" 1
"h" 7
```

```
"i" 9
"k" 1
"l" 4
"m" 3
"n" 9
```

```
"i" 9
"k" 1
"l" 4
"m" 3
"n" 9
```

```
"o" 10
"q" 1
"r" 9
"s" 7
"t" 11
```

```
"u" 3
"v" 1
"w" 2
"y" 2
```

Un ejemplo más realista

Ejecuciones

accidentes de circulación

cirrosis etc.

Partimos de una tabla en formato csv con los datos de causas de muerte por países y años:

Cód. país

annual-number														
ARCHIVO INICIO INSERTAR DISEÑO DE PÁGINA FÓRMULAS DATOS REVISAR VISTA FOXIT PDF ACROBAT														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Entitv	Code	Year	Executi	Meningi	Lower resc	Intestir	Protein-	Terrori	Cardiovascul	Dementia /d	Kidnev disea	Respiratorv	Liver dise
450	United States	USA	2007	42	1538	80095	26	3074	0	827191	197751	61820	161312	482
451	United States	USA	2008	37	1519	81095	26	3185	2	823970	204359	64807	166006	501
452	United States	USA	2009	52	1474	81044	25	3252	18	814684	209650	67415	168057	518
453	United States	USA	2010	46	1401	80847	25	3349	4	805696	216503	70175	169342	532
454	United States	USA	2011	43	1390	82789	26	3511	0	817311	224952	73837	175152	552
455	United States	USA	2012	43	1360	83685	26	3635	7	821110	231700	76101	177968	562
456	United States	USA	2013	39	1359	85776	27	3807	23	830227	238418	78992	182150	582
457	United States	USA	2014	35	1347	87280	27	4021	26	840356	244051	81192	185238	602
458	United States	USA	2015	28	1263	89325	28	4241	54	857258	249846	82242	189604	612
459	United States	USA	2016	20	1408	91658	29	4230	68	880573	254206	83665	192699	618
460	United States	USA	1990		2769	67896	30	1534	5	875093	125258	30479	98675	358
461	United States	USA	1991		2366	69158	30	1575	2	873846	129813	31213	102697	352
462	United States	USA	1992		2319	69914	30	1626	2	868099	133985	31891	105853	352
463	United States	USA	1993		2291	72047	30	1750		892670	140921	22622	112697	352
ted States;USA;2016;20;1408;91658;29;4230;68;880573;254206;83665;192699														
ted States;USA;1990;;2769;67896;30;1534;5;875093;125258;30479;98675;358														
467	United States	USA	1997		2285	78363	29	2276	2	900520	156269	38760	128119	360
468	United States	USA	1998		2157	79710	29	2466	4	902631	159524	40651	132967	362
469	United States	USA	1999		2030	79322	29	2697	20	913581	164560	43457	139986	362
470	United States	USA	2000		1938	79873	29	2829	0	911396	168177	45703	143901	372

United States;USA;2016;20;1408;91658;29;4230;68;880573;254206;83665;192699;61821; ...
United States;USA;1990;;2769;67896;30;1534;5;875093;125258;30479;98675;35809; ...

¿Cuántas muertes ha habido en Estados Unidos (USA, segunda columna de la tabla) debidas a ejecuciones (cuarta columna en la tabla)?

Un ejemplo más realista

Partimos de una tabla en formato csv con los datos de causas de muerte por países y años:

¿Cuántas muertes ha habido en Estados Unidos (USA, segunda columna de la tabla) debidas a ejecuciones (cuarta columna e `def mappero(self, linea):` línea):

	A	B	C	D											M	N	
1	Entiv	Code	Year	Executi Mei											es Respiratorv (Liver dise	
450	United States	USA	2007	42											20	161312	482
451	United States	USA	2008	37											07	166006	501
452	United States	USA	2009	52											15	168057	518
453	United States	USA	2010	46											75	169342	532
454	United States	USA	2011	43											37	175152	552
455	United States	USA	2012	43											01	177968	566
456	United States	USA	2013	39	1359	85776	27	3807	23	830227	238418	78992	182150	582			
457	United States	USA	2014	35	1347	87280	27	4021	26	840356	244051	81192	185238	600			
458	United States	USA	2015	28	1262	89225	28	4241	54	857258	248046	82242	188604	611			
459	United States	USA	2016	20	1408	91658	29	4230	68	880573	254206	83665	192699	618			
460	United States	USA	1990		2769	67896	30	1534	5	875093	125258	30479	98675	358			
461	United States	USA	1991		2366	69158	30	1575	2	873846	129813	31213	102697	355			
462	United States	USA	1992		2319	69914	30	1626	2	868099	133985	31891	105853	352			
463	United States	USA	1993		2381	70647	30	1750		892670	140931	32633	112687	359			
ted States;USA;2016;20;1408;91658;29;4230;68;880573;254206;83665;192699																	
ted States;USA;1990;;2769;67896;30;1534;5;875093;125258;30479;98675;358																	
467	United States	USA	1997		2285	78363	29	2276	2	900520	156269	38760	128119	360			
468	United States	USA	1998		2157	79710	29	2466	4	902631	159524	40651	132967	362			
469	United States	USA	1999		2030	79322	29	2697	20	913581	164560	43457	139986	367			
470	United States	USA	2000		1938	79873	29	2829	0	911396	168177	45703	143901	370			

United States;USA;2016;20;1408;91658;29;4230;68;880573;254206;83665;192699;61821; ...

United States;USA;1990;;2769;67896;30;1534;5;875093;125258;30479;98675;35809;...

Un ejemplo más realista

Partimos de una tabla en formato csv con los datos de causas de muerte por países y años:

¿Cuántas muertes ha habido en Estados Unidos (USA, segunda columna de la tabla) debidas a ejecuciones (cuarta columna de la tabla)?

	A	B	C	D
	Entity	Code	Year	Execution
450	United States	USA	2007	42
451	United States	USA	2008	37
452	United States	USA	2009	52
453	United States	USA	2010	46
454	United States	USA	2011	43
455	United States	USA	2012	43
456	United States	USA	2013	39
457	United States	USA	2014	35
458	United States	USA	2015	28
459	United States	USA	2016	20
460	United States	USA	1990	
461	United States	USA	1991	
462	United States	USA	1992	
463	United States	USA	1993	

```
def mapper(self, _, linea):
    campos = linea.split(";")
    cod_pais = campos[1]
    num_ejecs = int(campos[3])
    if cod_pais == "USA":
        yield "ejecs", num_ejecs
```

```
def reducer(self, key, valores):
    yield key, sum(valores)
```

```
def to_int(cadena):
    try:
        return int(cadena)
    except:
        return 0
```

```
def mapper(self, _, linea):
    campos = linea.split(";")
    cod_pais = campos[1]
    num_ejecs = to_int(campos[3])
    if cod_pais == "USA":
        yield "ejecs", num_ejecs
```

United States;USA;2016;20;1408;91658;29;4230;68;880573;254206;8;

United States;USA;1990;;2769;67896;30;1534;5;875093;125258;30479;98675;35809;...

467	United States	USA	1997		2285	78363	29	2276	2	900520	156269	38760	128119	36
468	United States	USA	1998		2157	79710	29	2466	4	902631	159524	40651	132967	36
469	United States	USA	1999		2030	79322	29	2697	20	913581	164560	43457	139986	36
470	United States	USA	2000		1938	79873	29	2829	0	911396	168172	45703	143901	37

Un ejemplo más realista y general

¿Cuántas muertes ha habido en Estados Unidos (USA, segunda columna de la tabla) debidas a ejecuciones (cuarta columna en la tabla)?

```
! python ".\death_cause.py" ".\annual-number-of-deaths-by-cause.csv" -q
```

```
"ejecs" 385
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Entiv	Code	Year	Executi	Meningi	Lower resc	Intestir	Protein-	Terrori	Cardiovascul	Dementia (d	Kidnev disea	Respiratorv	Liver dis
150	United States	USA	2007	42	1538	80095	26	3074	0	827191	197751	61820	161312	48
151	United States	USA	2008	37	1519	81095	26	3185	2	823970	204359	64807	166006	50
152	United States	USA	2009	52	1474	81044	25	3252	18	814684	209650	67415	168057	51
153	United States	USA	2010	46	1401	80847	25	3349	4	805696	216503	70175	169342	51
154	United States	USA	2011	43	1390	82789	26	3511	0	817311	224952	73837	175152	55
155	United States	USA	2012	43	1360	83685	26	3635	7	821110	221700	76101	177968	54

```
! python ".\death_cause_country.py" --country=USA ".\annual-number-of-deaths-by-cause.csv" -q
```

```
"USA" 385
```

```
! python ".\death_cause_country.py" --country=ESP ".\annual-number-of-deaths-by-cause.csv" -q
```

```
"ESP" 0
```

Un ejemplo más realista y general

¿Cuántas muertes ha habido en Estados Unidos (USA, segunda columna de la tabla) debidas a ejecuciones (cuarta columna en la tabla)?

```
class MRCauseOfDeath(MRJob):
    def configure_args(self):
        super(MRCauseOfDeath, self).configure_args()
        self.add_passthru_arg( '--country' default='Spain', help="Indica el código del país.")

    def mapper(self, _, linea):
        campos = linea.split(",")
        cod_pais= campos[1]
        if cod_pais == self.options.country:
            num_ejec = to_int(campos[3])
            yield cod_pais, num_ejes

    def reducer(self, key, valores):
        yield key, sum(valores)
```

<https://mrjob.readthedocs.io/en/latest/guides/writing-mrjobs.html>

<https://stackoverflow.com/questions/66734113/is-it-possible-to-pass-arguments-to-mr-job>

```
! python ".\death_cause_country.py" --country=ESP ".\annual-number-of-deaths-by-cause.csv" -q
```

```
"ESP" 0
```

Propiedades matemáticas asumidas

- Suma

$[1, 2, 3, 4, 5] \rightarrow [1, 2, 3], [4, 5] \rightarrow [6, 9] \rightarrow 15$
 $[1, 2, 3, 4, 5] \rightarrow [1, 2], [3, 4, 5] \rightarrow [3, 12] \rightarrow 15$
...

- Media

$[1, 2, 3, 4, 5] \rightarrow [1, 2, 3], [4, 5] \rightarrow [2, 4.5] \rightarrow 3,25$
 $[1, 2, 3, 4, 5] \rightarrow [1, 2], [3, 4, 5] \rightarrow [1.5, 4] \rightarrow 2,75$

La operación reduce ha de ser asociativa y conmutativa.

Si no están estas propiedades garantizadas, se ha de emplear otro procedimiento.

V. ej3 - media - postprocesamiento

Media: ejemplo con post-procesamiento

Tenemos un conjunto de colores:

Colores = {Azul, Blanco, Verde, Amarillo}

Cada color tiene asociado un valor entero:

Valores = {"Azul": 3.0, "Blanco": 5.0,
 "Verde": 15.0, "Amarillo": 10.0}

Elegimos un color, con igual probabilidad, entre la lista Colores, y para cada color, generamos un número real, extraído uniformemente en el intervalo del valor del color, ± 1 . Por ejemplo, si el color elegido es el azul, el valor será un número real del intervalo $[3 - 1, 3 + 1]$. He aquí una muestra:

Deseamos calcular la media de cada color.

datos.txt

```
Amarillo 9.48
Azul 2.18
Blanco 4.71
Verde 14.4
Azul 3.32
Verde 15.27
Amarillo 10.72
Verde 15.96
Azul 2.19
Verde 14.52
Amarillo 10.04
Blanco 5.93
Azul 2.41
Blanco 4.61
Amarillo 9.1
Blanco 4.35
Blanco 5.52
Azul 3.14
Azul 3.3
Verde 14.77
```

Media: ejemplo con post-procesamiento

datos.txt

Amarillo 9.48
Azul 2.18
Blanco 4.71
Verde 14.4
Azul 3.32
Verde 15.27
Amarillo 10.72
Verde 15.96
Azul 2.19
Verde 14.52
Amarillo 10.04
Blanco 5.93
Azul 2.41
Blanco 4.61
Amarillo 9.1
Blanco 4.35
Blanco 5.52
Azul 3.14
Azul 3.3
Verde 14.77

```
1 import sys
2 from mrjob.job import MRJob
3
4 def suma_doble(pares):
5     """Ej. [(1, 10), (2, 20), (3, 30)] --> (6, 60)"""
6     a, b = 0, 0
7     for x, y in pares:
8         a, b = a + x, b + y
9     return a, b
10
11 class MRSumaTotales(MRJob):
12
13     def mapper(self, _, linea):
14         [color, x] = linea.split()
15         yield color, (float(x), 1)
16
17     def reducer(self, key, values):
18         yield key, suma_doble(values)
19
20 if __name__ == '__main__':
21     archivo_datos = sys.argv[1]
22     trabajo = MRSumaTotales(args=[archivo_datos])
23     with trabajo.make_runner() as runner:
24         runner.run()
25         for key, value in trabajo.parse_output(runner.cat_output()):
26             media = value[0] / value[1]
27             media_str = str(round(media, 2))
28             print(key + " - " + media_str)
```

```
! python medias.py datos.txt -q
```

Amarillo - 9.98
Azul - 3.02
Blanco - 4.93
Verde - 15.06

Observaciones

- Documentación de las funciones omitida por brevedad en la explicación
- Código funcional completo en los script, con pruebas de funcionamiento
- Algunas funciones no usadas en el código, para propiciar otras pruebas

Bibliografía

- MapReduce, entrada en la Wikipedia:
<https://es.wikipedia.org/wiki/MapReduce>
- MapReduce tutorial, en la web oficial de Hadoop:
https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html