Spark in Python

Running a Pyspark script

- In pyhton spark is an object stored in a variable
  - Called : Spark Context (sc)
- Interactive mode : it's already there

```
$pyspark
>>>print sc
<pyspark.context.SparkContext object at 0x386cb90>
```

- If you have a pyspark script : You have to create it

```
#In the script
from pyspark import SparkContext
sc = SparkContext(appName="Simple App")
.
.
$spark-submit script.py
```

Interactive mode: Example(1)

```
$pyspark
>>>temp=range(10000)
>>>temp_rdd=sc.parallelize(temp)
>>>top10=temp_rdd.top(10)
>>>print top10


>>>def filt(x):
... return x%2==0

>>>temp_rdd2=temp_rdd.filter(filt)
>>>top10=temp_rdd2.top(10)
>>>print top10

>>>all=temp_rdd2.collect()
>>>print all
```

Interactive mode: Example(2)

- Compute 10000 random numbers
  - Between 1 and 10:
  - Random seed : all the numbers between 0 and 9999
  - return the seed and the random number

```
>>>import random
>>>def newrand(x):
... random.seed(x)
... return (x,random.randint(1,10))

>>>rand_rdd=temp_rdd.map(newrand)
>>>print rand_rdd.top(10)
```

Interactive mode: Example(3)

- Compute 10000 random numbers
  - Between 1 and 10:
  - Random seed : all the numbers between 0 and 9999
  - return the seed and the random number

```
>>>import random
>>>def newrand(x):
... random.seed(x)
... return (x,random.randint(1,10))

>>>rand_rdd=temp_rdd.map(newrand)
>>>print rand_rdd.top(10)
```

Count even per 1000

```
>>>def filt2(x):
... return x[1]%2==0

>>>grouped=rand_rdd.filter(filt2).groupByKey()

>>>print grouped.top(1)
#not the datatypes we would expect

#there is a faster way to define a quick function
>>>grouped=grouped.map(lambda x:(x[0],list(x[1])))
>>>print grouped

>>>countPG=grouped.map(lambda x: (x[0],len(x[1])))
>>>print countPG
```

10 random per seed

```
>>>def newrand2(x):
... random.seed(x)
... return (x,[random.randint(0,10) for i in range(1,10)])

>>>rnd2=temp_rdd.map(newrand2).map(convertkey)
#we can flatten each array
>>>print rnd2=temp_rdd.map(newrand2).map(convertkey)

#there other ways to aggregate per key
>>>def countEvens(x):
... count=0
... for v in x[1]:
...  if v%2==0 : count+=1
... return (x,count)

>>> countPG=rnd2.map(countEvens).
       reduceByKey(lambda x,y : x+y).collect()
```

Sharing a variable

- For the same 10 seeds produce 10000 random numbers
  - 1000 times 10 random numbers

```
>>>seed_numbers=range(10)
>>>sd=sc.broadcast(seed_numbers)

>>>def newrand2(x):
... result=[]
... for v in sd.value:
...  random.seed(v)
...  result.append(random.randint(1,10))
... return result

>>> rdd=sc.parallelize(range(1000)).map(newrand2)
```

Other Functions

join e.g x.join(y)	returns tuples of all pairs that can be joind on the same key
count	size of an rdd
max/min	
sum/mean/stdev	
saveAsXXX	Where XXX =[TextFile,PicleFile,HadoopFile)
countByKey	
combineByKey	combine the elements for each key with a custom function
aggregateByKey	Aggregate the values of each key, with a custom function(s) and a neutral "zero value".

aggregateByKey(*zeroValue, seqFunc, combFunc****)***

- zeroValue: starting value
- seqFunc: how to combine data and RDD types
- combFunc: how to combine final data

```
#there other ways to aggregate per key
>>>def combine(x,y):
...  x.extend(list(y))
...  return x

>>> countPG=rnd2.aggregateByKey([],combine,combine).
       map(countEvens).collect()
```

What about in a script?

- Broadcasted variables are not necessarily seen by functions
  - Functions have to take as an argument extra variables (broadcasted)
    - How to pass the extra variable in maps e.t.c.?
    - Use Python partial
  - See file example.py

THE END