Una de las ventajas de trabajar con Spark son las consolas interactivas que tiene para dos de los lenguajes con los que se puede programar, Scala (que se ejecuta en una máquina virtual Java- JVM) y Python. Estas consolas permiten analizar los datos de forma interactiva, con la conexión a los clústeres.

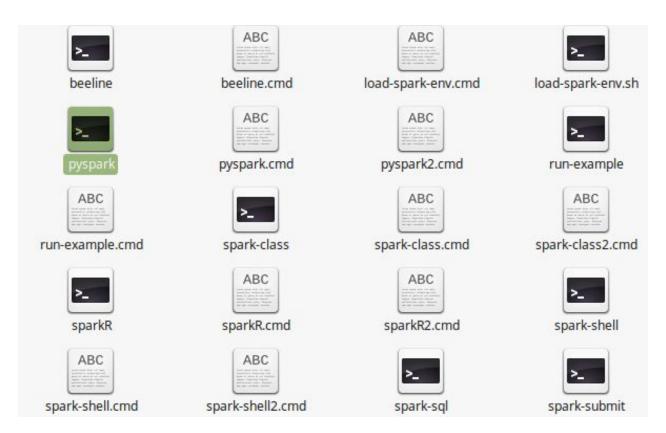
La ejecución de comandos en spark se puede hacer de 2 formas, escribiendo un programa completo y ejecutarlo con el script **sparkSubmit** o bien mediante la consola interactiva de python.

Cuando trabajamos directamente con la shell, es en sí misma el programa principal , y se escriben de una en una las operaciones que se quieran aplicar.

El SparkContext se crea automáticamente con la variable llamada sc

Ejecutamos Pyspark:

\$./bin/pyspark



```
jmocejmoc-HP-Compaq-6005-Pro-SFF-PC ~/Descargas/spark-2.0.2-bin-hadoop2.7/bin $ ./pyspark
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/04/16 16:58:22 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/04/16 16:58:23 WARN Utils: Your hostname, jmoc-HP-Compaq-6005-Pro-SFF-PC resolves to a
loopback address: 127.0.1.1; using 192.168.100.11 instead (on interface wlx0012fb2961d5)
17/04/16 16:58:23 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Welcome to

/// _____//__//__/ version 2.0.2

Using Python version 2.7.12 (default, Nov 19 2016 06:48:10)
SparkSession available as 'spark'.
>>>
```

El primer paso es crear un objeto de SparkContext, con la variable sc. Esta variable ya está declarada por defecto en la consola interactiva por lo que basta con escribir sc en la consola para ver que el objeto SparkContext ya está creado

Una vez declarado, creamos el RDD, con sc.textFile declaramos las líneas de un fichero de texto y mediante funciones lambda podemos contar el número de palabras del fichero de texto

```
>>> lines = sc.textFile("README.md");
>>> counts = lines.flatMap(lambda x:
x.split(' ')).map(lambda x:
(x,1)).reduceByKey(lambda x,y: x+y);
>>> print counts.collect();
```