Spark Context

SparkContext is the internal engine that allows the connections with the clusters. If you want to run an operation, you need a SparkContext.

Create a SparkContext

First of all, you need to initiate a SparkContext.

```
import pyspark

from pyspark import SparkContext

sc =SparkContext()
```

Now that the SparkContext is ready, you can create a collection of data called RDD, Resilient Distributed Dataset. Computation in an RDD is automatically parallelized across the cluster.

```
nums= sc.parallelize([1,2,3,4])
```

You can access the first row with take

```
nums.take(1)
```

You can apply a transformation to the data with a lambda function. In the example below, you return the square of nums. It is a map transformation

```
squared = nums.map(lambda x: x*x).collect()

for num in squared:

    print('%i ' % (num))
```

SQLContext

A more convenient way is to use the DataFrame. SparkContext is already set, you can use it to create the dataFrame. You also need to declare the SQLContext

SQLContext allows connecting the engine with different data sources. It is used to initiate the functionalities of Spark SQL.

```
from pyspark.sql import Row

from pyspark.sql import SQLContext

sqlContext = SQLContext(sc)
```

Let's create a list of tuple. Each tuple will contain the name of the people and their age. Four steps are required:

Step 1) Create the list of tuple with the information

```
list_p=[('John',19),('Smith',29),('Adam',35),('Henry',50)]
```

Step 2) Build a RDD

```
rdd = sc.parallelize(list_p)
```

Step 3) Convert the tuples

```
rdd.map(lambda x: Row(name=x[0], age=int(x[1])))
```

Step 4) Create a DataFrame context

```
sqlContext.createDataFrame(ppl)
```

```
list_p = [('John',19),('Smith',29),('Adam',35),('Henry',50)]

rdd = sc.parallelize(list_p)

ppl = rdd.map(lambda x: Row(name=x[0], age=int(x[1])))

DF_ppl = sqlContext.createDataFrame(ppl)
```

If you want to access the type of each feature, you can use printSchema()

```
DF_ppl.printSchema()
```