# Hadoop

Stores and processes data in a master slave setup.

Also provides recommendation engine, collective intelligence

SPARK - processing tool written in scala. Hadoop is uses it for processing the data.

# Scala

Has Persistent data structure (similar to pointers)

Has a REPL(Read Evaluate Print Loop) similar to that of Python

### Variables

Mutable variables - "var”

Immutable variables - "val" (Similar to final in java). Preference is to use immutable vals.

Scala has type inference and is statically typed (not dynamically typed)  
e.g.

***var a = 5***

***a = “Hi”***

This will throw error in scala and the type inference is done statically (first time initialized). But this will work in python and javascript where type inference is dynamic (integer will change to String)

### String Operations

== can be used to compare two strings (unlike Java where it’s not safe)

Interpolation

***s”$greeting, $name, Welcome”***

In this greeting and name are variables.

s” is for string interpolation

f” is for number interpolation

### Emptiness

Any (parent)

AnyVal (Char, Double, Int , etc). Can’t be null

AnyRef (Collections, String, Classes, etc). Can be null

**null:**  
Values cant be null, but References can be null

**Null:**

Null is a trait. null is type of Null. AnyRef can be type of Null

**Nothing:**

It cannot be instantiated. Every type extends Nothing

**Nil:**

The last element of List is Nil. Lists are internally represented as Doubly Linked List

**None**:

Option is a collection of value Some or None. These are wrappers. None is retuned when we have no valu to return when wrapped around Option.

**Unit:**

Return type of function which returns nothing.

### Functions vs Methods

**Method:**

These are not objects

When def is the signature, it’s a method.

Method arguments need to be typed explicitly. But the return type need not be typed explicitly.

The last line of the method is always returned. There is no return statement.

Can’t be used in pipeline.

Faster and better performing.

Can be converted to function.

**Function:**

These are objects.

So these are of certain traits like function1 (when 1 argument is given), function2 (when 2 arguments are given), etc. (max 23 arguments are only allowed).

Can be used in pipeline.

Function can be returned from another function. It can take a function as an argument too. A val or var can store a function.

Can’t be converted to method. By either of below two ways

val functionName: (methodInputType) => methodReturnType = methodName

val functionName = methodName \_

(underscore is a placeholder for method signature)

### .tupled can be used here.

### Expression vs Statements

Something that returns it’s a expression. Something that doesn’t return anything is a statement.

Named expression blocks are functions. If name is not there, it becomes an anonymous function (using lambda)