Slide 10:
Passing functions to Spark. I wanted to touch a little bit on this before we move further. This is important to understand as you begin to think about the business logic of your application.

The design of Spark's API relies heavily on passing functions in the driver program to run on the cluster. When a job is executed, the Spark driver needs to tell its worker how to process the data.

There are three methods that you can use to pass functions.

The first method to do this is using an anonymous function syntax. You saw briefly what an anonymous function is in the first lesson. This is useful for short pieces of code. For example, here we define the anonymous function that takes in a particular parameter x of type Int and add one to it. Essentially, anonymous functions are useful for one-time use of the function. In other words, you don't need to explicitly define the function to use it. You define it as you go. Again, the left side of the => are the parameters or the arguments. The right side of the => is the body of the function.

Another method to pass functions around Spark is to use static methods in a global singleton object. This means that you can create a global object, in the example, it is the object MyFunctions. Inside that object, you basically define the function func1. When the driver requires that function, it only needs to send out the object – the worker will be able to access it. In this case, when the driver sends out the instructions to the worker, it just has to send out the singleton object.

It is possible to pass reference to a method in a class instance, as opposed to a singleton object. This would require sending the object that contains the class along with the method. To avoid this consider copying it to a local variable within the function instead of accessing it externally.

Example, say you have a field with the string Hello. You want to avoid calling that directly inside a function as shown on the slide as x => field + x.

Instead, assign it to a local variable so that only the reference is passed along and not the entire object shown val field_ = this.field.

For an example such as this, it may seem trivial, but imagine if the field object is not a simple text Hello, but is something much larger, say a large log file. In that case, passing by reference will have greater value by saving a lot of storage by not having to pass the entire file.

Slide 11:
Back to our regularly scheduled program.

At this point, you should know how to link dependencies with Spark and also know how to initialize the SparkContext. I also touched a little bit on passing functions with Spark to give you a better view of how you can program your business logic. This course will not focus too much on how to program business logics, but there are examples available for you to see how it is done. The purpose is to show you how you can create an application using a simple, but effective example which demonstrates Spark's capabilities.

Once you have the beginning of your application ready by creating the SparkContext object, you can start to program in the business logic using Spark's API available in Scala, Java, or Python. You create the RDD from an external dataset or from an existing RDD. You use transformations and actions to compute the business logic. You can take advantage of RDD persistence, broadcast variables and/or accumulators to improve the performance of your jobs.

Here's a sample Scala application. You have your import statement. After the beginning of the object, you see that the SparkConf is created with the application name. Then a SparkContext is created. The several lines of code after is creating the RDD from a text file and then performing the Hdfs test on it to see how long the iteration through the file

takes. Finally, at the end, you stop the SparkContext by calling the stop() function.

Again, just a simple example to show how you would create a Spark application. You will get to practice this in the lab exercise.

Slide 12:
I mentioned that there are examples available which shows the various usage of Spark. Depending on your programming language preference, there are examples in all three languages that work with Spark. You can view the source code of the examples on the Spark website or within the Spark distribution itself. I provided some screenshots here to show you some of the examples available.

On the slide, I also listed the step to run these examples. To run Scala or Java examples, you would execute the run-example script under the Spark's home/bin directory. So for example, to run the SparkPi application, execute run-example SparkPi, where SparkPi would be the name of the application. Substitute that with a different application name to run that other application.

To run the sample Python applications, use the spark-submit command and provide the path to the application.