SCALA FUNCTIONS

A function is a group of statements that together perform a task. You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically, the division usually is so that each function performs a specific task.

Scala has both functions and methods and we use the terms method and function interchangeably with a minor difference. A Scala method is a part of a class which has a name, a signature, optionally some annotations, and some bytecode where as a function in Scala is a complete object which can be assigned to a variable. In other words, a function, which is defined as a member of some object, is called a method.

A function definition can appear anywhere in a source file and Scala permits nested function definitions, that is, function definitions inside other function definitions. Most important point to note is that Scala function's name can have characters like +, ++, ~, &,-, -- , \, /, : etc.

Function Declarations:

A scala function declaration has the following form:

```
def functionName ([list of parameters]) : [return type]
```

Methods are implicitly declared *abstract* if you leave off the equals sign and method body. The enclosing type is then itself abstract.

Function Definitions:

A scala function definition has the following form:

```
def functionName ([list of parameters]) : [return type] = {
   function body
   return [expr]
}
```

Here, **return type** could be any valid scala data type and **list of parameters** will be a list of variables separated by comma and list of parameters and return type are optional. Very similar to Java, a **return** statement can be used along with an expression in case function returns a value. Following is the function which will add two integers and return their sum:

```
object add{
   def addInt( a:Int, b:Int ) : Int = {
      var sum:Int = 0
      sum = a + b

      return sum
   }
}
```

A function, which does not return anything, can return **Unit** which is equivalent to **void** in Java and indicates that function does not return anything. The functions which do not return anything in Scala, they are called procedures. Following is the syntax

```
object Hello{
   def printMe( ) : Unit = {
      println("Hello, Scala!")
   }
}
```

Calling Functions:

Scala provides a number of syntactic variations for invoking methods. Following is the standard

way to call a method:

```
functionName( list of parameters )
```

If function is being called using an instance of the object then we would use dot notation similar to Java as follows:

```
[instance.]functionName( list of parameters )
```

Following is the final example to define and then call the same function:

```
object Test {
   def main(args: Array[String]) {
        println( "Returned Value : " + addInt(5,7) );
   }
   def addInt( a:Int, b:Int ) : Int = {
      var sum:Int = 0
      sum = a + b

      return sum
   }
}
```

When the above code is compiled and executed, it produces the following result:

```
C:/>scalac Test.scala
C:/>scala Test
Returned Value : 12

C:/>
```

Scala functions are the heart of Scala programming and that's why Scala is assumed as a functional programming language. Following are few important concepts related to Scala functions which should be understood by a Scala programmer.

Functions Call-by-Name	Functions with Named Arguments
Function with Variable Arguments	Recursion Functions
Default Parameter Values	Higher-Order Functions
Nested Functions	Anonymous Functions
Partially Applied Functions	Currying Functions