# ADVANCED FUNCTIONAL THINKING

DR SURBHI SARASWAT
SCHOOL OF COMPUTER SCIENCE

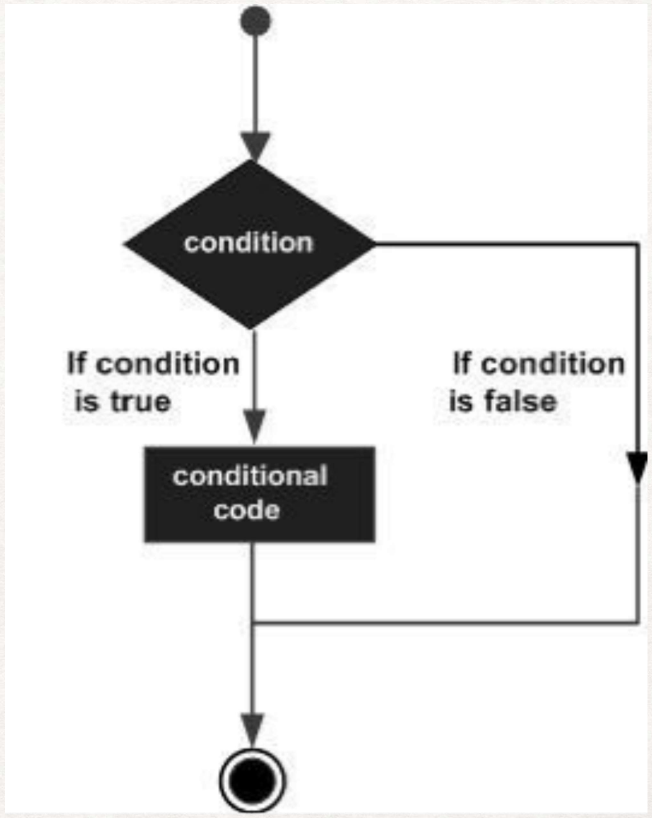
## SCALA IF... ELSE STATEMENTS

- The **if** Statement:
- SYNTAX:

```
if(Boolean_expression)
{
// Statements will execute if the Boolean expression is true
}
```

- Example:

```
object Test {
        def main(args: Array[String]) {
            var x = 10;
            if( x < 20 ){
                println("This is if statement");
            }
        }
    }
```



# SCALA IF... ELSE STATEMENTS

- The **if...else** Statement:

Syntax:

```
if(Boolean_expression){
   //Executes when the Boolean expression is true
}else{
   //Executes when the Boolean expression is false
}
```

- The **if...else if...else** Statement:

SYNTAX:

```
if(Boolean_expression 1){
   //Executes when the Boolean expression 1 is true
} else if(Boolean_expression 2){
   //Executes when the Boolean expression 2 is true
} else {
   //Executes when the none of the above condition is true.
}
```

EXAMPLE:

```
object Test {
    def main(args: Array[String]) {
        var x = 30;
        if( x < 20 ){
           println("This is if statement");
         }    else{
          println("This is else statement");
         }
    }
}
```

## SCALA IF... ELSE STATEMENTS

**EXAMPLE: if..else if..else**

```
object Test {
    def main(args: Array[String]) {
        var x = 30;
        if( x == 10 ){
            println("Value of X is 10");
}      else if( x == 20 ){
           println("Value of X is 20");
}     else if( x == 30 ){
           println("Value of X is 30");
}     else{
          println("This is else statement");
          }
    }
}
```

**EXAMPLE: Nested if**

```
object Test {
    def main(args: Array[String]) {
        var x = 30;
        var y = 10;

        if( x == 30 ){
            if( y == 10 ){
            println("X = 30 and Y = 10");
             }
        }
    }
}
```

## SCALA LOOP TYPES

- **for** loop
    - The **for** Loop with Ranges

The simplest syntax of a **for** loop in Scala is:

```
for( var x <- Range ){
statement(s); }
```

Here, the **Range** could be a range of numbers and that is represented as **i to j** or sometime like **i until j**. The left-arrow <- operator is called a *generator*, so named because it's generating individual values from a range.

```
object Test {
    def main(args: Array[String]) {
        var a = 0;
        for( a <- 1 to 10){                    // for loop execution with a range
            println( "Value of a: " + a );
        }
    }
}
```

```
output:
1
2
3
4
5
6
7
8
9
10
```

## SCALA LOOP TYPES

You can use multiple ranges separated by semicolon (;) within a **for loop** and in that case loop will iterate through all the possible computations of the given ranges.

Following is an example of using just two ranges, you can use more than two ranges as well.

```
object Test {
    def main(args: Array[String]) {
        var a = 0;
        var b = 0;
        for( a <- 1 to 3; b <- 1 to 3){          // for loop execution with a range
            println( "Value of a: " + a );
            println( "Value of b: " + b );
        }
    }
}
```

## SCALA LOOP TYPES

- The **for** Loop with **Collections**
  - The syntax of a for loop with a collection is as follows:

```
for( var x <- List ){
        statement(s); }
```

- Here, the **List** variable is a collection type having a list of elements and *for loop* iterates through all the elements returning one element in x variable at a time.

Example:

```
object Test {
   def main(args: Array[String]) {
   var a = 0;
   val numList = List(1,2,3,4,5,6);
   for( a <- numList ){             // for loop execution with a collection
         println( "Value of a: " + a );
    }
   }
}
```

## SCALA LOOP TYPES

- The **for** Loop with **Filters**
  - Scala's for loop allows to filter out some elements using one or more **if** statement(s). Following is the syntax of *for loop* along with filters.

```
for( var x <- List  if condition1; if condition2... )    {
   statement(s);   }
```

To add more than one filter to a for expression, separate the filters with semicolons(;).

```
object Test {
   def main(args: Array[String]) {
   var a = 0;
   val numList = List(1,2,3,4,5,6,7,8,9,10);
   for( a <- numList  if a != 3; if a < 8  ){          // for loop execution with multiple filters
         println( "Value of a: " + a );
         }
    }
}
```

```
output:
1
2
4
5
6
7
```

## SCALA LOOP TYPES

- the **for** loop along with **yield**

```
object Test {
   def main(args: Array[String]) {
      var a = 0;
      val numList = List(1,2,3,4,5,6,7,8,9,10);
      // for loop execution with a yield
      var retVal = for{ a <- numList if a != 3; if a < 8 } yield a
      // Now print returned values using another loop.
      for( a <- retVal){
         println( "Value of a: " + a );
      }
   }
}
```

## STANDARD INPUT

- scala.io.StdIn.readLine()
- scala.io.StdIn.readInt()
- scala.io.StdIn.readFloat()
- Example:

```
object ReadInputExample {
   def main(args: Array[String]) {
      print("Enter a number:  ")
      var hours = scala.io.StdIn.readInt()
      hours = hours + 1
      println("Your entry + 1 :  "+hours)
   }
}
```

## RANDOM NUMBER

- `val rand = new scala.util.Random`

- **Example:**

```
scala> rand.nextInt()
val res1: Int = 2082034767

scala> rand.nextInt(50)
val res4: Int = 19


scala> rand.between(3, 15)
val res6: Int = 9
```