# Loops

## Scala while loop

In Scala, while loop is used to iterate code till the specified condition. It tests boolean expression and iterates again and again. You are recommended to use while loop if you don't know number of iterations prior.

Syntax

```
while(boolean expression){
   // Statements to be executed
}
```

## Scala do-while loop

example

```
object MainObject {
  def main(args: Array[String]) {
      var a = 10;        // Initialization
      do {
         println( a );
         a = a + 2;     // Increment
      }
      while( a <= 20 )    // Condition
  }
}
```

# Scala for loop

In scala, for loop is known as for-comprehensions. It can be used to iterate, filter and return an iterated collection. The for-comprehension looks a bit like a for-loop in imperative

languages, except that it constructs a list of the results of all iterations.

Syntax

```
for( i <- range){
   // statements to be executed
}
```

In the above syntax, range is a value which has start and end point. You can pass range by using to or until keyword.

**Scala for-loop example by using to keyword**

```
object MainObject {
  def main(args: Array[String]) {
      for( a <- 1 to 10 ){
       println(a);
    }
  }
}
```

In the below example, until is used instead of to. The major difference between until and to is, to includes start and end value given in the range, while until excludes last value of the range. So, the below example will print only 1 to 9.

**Scala for-loop Example by using until keyword**

```
object MainObject {
  def main(args: Array[String]) {
      for( a <- 1 until 10 ){
       println(a);
```

```
    }
  }
}
```

It is helpful to apply until keyword when you are iterating string or array, because array range is 0 to n-1. until does not exceed to n-1. So, your code will not complain of upper range.

### Scala for-loop filtering Example

You can use for to filter your data. In the below example, we are filtering our data by passing a conditional expression. This program prints only even values in the given range.

```
object MainObject {
  def main(args: Array[String]) {
      for( a <- 1 to 10 if a%2==0 ){
       println(a);
    }
  }
}
```

### Scala for-loop Example by using yield keyword

In the above example, we have used yield keyword which returns a result after completing of loop iterations. The for use buffer internally to store iterated result and after finishing all iterations it yields the final result from that buffer. It does not work like imperative loop.

```
object MainObject {
  def main(args: Array[String]) {
      var result = for( a <- 1 to 10) yield a
      for(i<-result){
         println(i)
```

```
        }
    }
}
```