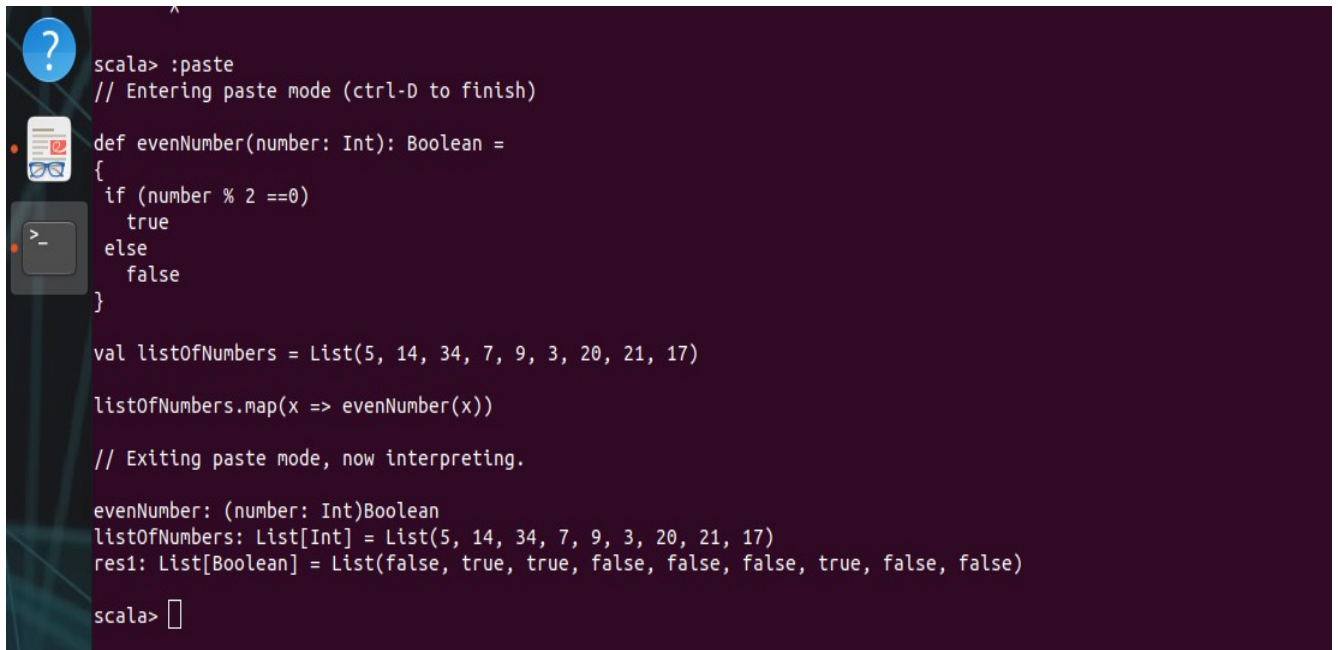


- Comme nous pouvons le voir avec l'image ci-dessous, nous avons défini une fonction permettant de dire si oui ou non un nombre est pair en retournant true ou false. Ensuite nous avons appliqué la fonction map avec notre fonction précédemment définie sur une liste d'entier.



```
scala> :paste
// Entering paste mode (ctrl-D to finish)

def evenNumber(number: Int): Boolean =
{
  if (number % 2 == 0)
    true
  else
    false
}

val listOfNumbers = List(5, 14, 34, 7, 9, 3, 20, 21, 17)

listOfNumbers.map(x => evenNumber(x))

// Exiting paste mode, now interpreting.

evenNumber: (number: Int)Boolean
listOfNumbers: List[Int] = List(5, 14, 34, 7, 9, 3, 20, 21, 17)
res1: List[Boolean] = List(false, true, true, false, false, false, true, false, false)

scala> 
```

- Extraction de la première et dernière de chaque chaîne d'une liste en utilisant la fonction map



```
scala> :paste
// Entering paste mode (ctrl-D to finish)

def firstCharacterOfString(chaine: String) = (chaine(0), chaine(chaine.length - 1))

val listOfString = List("Momo", "Polytechnic", "Vision", "Alpha", "Beta")

listOfString.map(x => firstCharacterOfString(x))

// Exiting paste mode, now interpreting.

firstCharacterOfString: (chaine: String)(Char, Char)
listOfString: List[String] = List(Momo, Polytechnic, Vision, Alpha, Beta)
res2: List[(Char, Char)] = List((M,o), (P,c), (V,n), (A,a), (B,a))

scala> 
```

- Chargement du contenu d'un fichier texte ensuite faire une itération sur chaque ligne du fichier:

```
momo@momo: ~  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
import scala.io.Source  
  
val lines = Source.fromFile("/home/momo/Desktop/Formation_Big_Data/file.txt").getLines.toList  
lines.map(x => println(x))  
  
// Exiting paste mode, now interpreting.  
Nous sommes en phase de test.  
Ceci est un fichier texte.  
Nous allons l'utiliser avec la fonction map.  
  
import scala.io.Source  
lines: List[String] = List(Nous sommes en phase de test., Ceci est un fichier texte., Nous allons l'utiliser avec la fonction map., "", "")  
res3: List[Unit] = List((), (), (), (), ())  
  
scala> 
```

- Exemples de concaténation de deux listes, d'ajout d'un nouvel élément dans une liste, d'union et d'intersection de liste

```
momo@momo: ~  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
val a = List(1,2,3)  
  
val b = List(4,5,6)  
List.concat(a, b)  
  
// Exiting paste mode, now interpreting.  
a: List[Int] = List(1, 2, 3)  
b: List[Int] = List(4, 5, 6)  
res4: List[Int] = List(1, 2, 3, 4, 5, 6)  
  
scala> 
```

```
momo@momo: ~  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
val listEle = List(4, 7, 3, 10, 35)  
  
listEle :+ 14  
  
// Exiting paste mode, now interpreting.  
listEle: List[Int] = List(4, 7, 3, 10, 35)  
res5: List[Int] = List(4, 7, 3, 10, 35, 14)  
  
scala> 
```

```
momo@momo: ~  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val listEle = List(4, 7, 3, 10, 35)  
val listEle1 = List(14, 78, 2, 4, 10, 68, 7, 12)  
  
listEle.intersect(listEle1)  
listEle.union(listEle1)  
  
// Exiting paste mode, now interpreting.  
  
listEle: List[Int] = List(4, 7, 3, 10, 35)  
listEle1: List[Int] = List(14, 78, 2, 4, 10, 68, 7, 12)  
res9: List[Int] = List(4, 7, 3, 10, 35, 14, 78, 2, 4, 10, 68, 7, 12)  
  
scala> 
```

Nous constatons que avec toutes ces opérations une nouvelle est retournée et la liste originale n'est pas modifiée.

- Transformation d'une liste en string avec la fonction reduce

```
momo@momo: ~  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
val listOfString = List("Momo", "Polytechnic", "Vision", "Alpha", "Beta")  
  
listOfString.reduce((x,y) => x + ";" + y)  
  
// Exiting paste mode, now interpreting.  
  
listOfString: List[String] = List(Momo, Polytechnic, Vision, Alpha, Beta)  
res10: String = Momo;Polytechnic;Vision;Alpha;Beta  
  
scala> 
```

- En Scala, un ListBuffer est semblable à un ArrayBuffer, sauf qu'il utilise une liste liée en interne au lieu d'un tableau. L'image ci-dessous montre un exemple d'utilisation d'un ArrayBuffer:

```
momo@momo: ~  
scala> :paste  
// Entering paste mode (ctrl-D to finish)  
  
import scala.collection.mutable.ArrayBuffer  
  
val nums = ArrayBuffer(71, 26, 54, 34, 12)  
  
nums += 87  
nums += 104  
nums += 4  
  
// Exiting paste mode, now interpreting.  
  
import scala.collection.mutable.ArrayBuffer  
nums: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(71, 26, 54, 34, 12, 87, 104, 4)  
res11: nums.type = ArrayBuffer(71, 26, 54, 34, 12, 87, 104, 4)
```

- Un Array en Scala un type particulier de collection en Scala. Il s'agit d'une structure de données de taille fixe qui stocke des éléments du même type de données. Les arrays sont assez similaires aux listes du point de vue où ils stockent des éléments de même type. Par contre les listes sont immuables, ce qui signifie que les éléments d'une liste ne peuvent pas être modifiés par affectation. De plus, les listes représentent une liste liée alors que les Arrays sont plats.
- Les vecteurs en Scala sont des structures de données immuables fournissant un accès aléatoire aux éléments et sont similaires aux listes. Ils sont beaucoup plus performants en matière d'accès aléatoire, en particulier dans les grandes collections. Ils offrent une performance très compétitive pour toutes les autres opérations courantes par rapport à toute autre collection immuable en Scala.