Les Collections en Scala

Table des matières

1. Les Séquences

```
scala> val colors = List("red", "blue", "green")
colors: List[java.lang.String] = List(red, blue, green)

scala> colors.head
res0: java.lang.String = red

scala> colors.tail
res1: List[java.lang.String] = List(blue, green)
```

```
scala> val fiveInts = new Array[Int](5)
fiveInts: Array[Int] = Array(0, 0, 0, 0, 0)
```

```
scala> val fiveToOne = Array(5, 4, 3, 2, 1)
fiveToOne: Array[Int] = Array(5, 4, 3, 2, 1)
```

```
scala> fiveInts(0) = fiveToOne(4)

scala> fiveInts
res1: Array[Int] = Array(1, 0, 0, 0, 0)
```

```
scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer

scala> val buf = new ListBuffer[Int]
buf: scala.collection.mutable.ListBuffer[Int] = ListBuffer()

scala> buf += 1

scala> buf += 2

scala> buf
res11: scala.collection.mutable.ListBuffer[Int]
  = ListBuffer(1, 2)

scala> buf.toList
res13: List[Int] = List(1, 2)
```

```
scala> import scala.collection.mutable.ArrayBuffer
import scala.collection.mutable.ArrayBuffer
```

```
scala> val buf = new ArrayBuffer[Int]()
buf: scala.collection.mutable.ArrayBuffer[Int] =
  ArrayBuffer()
```

```
scala> buf += 12

scala> buf += 15

scala> buf
res16: scala.collection.mutable.ArrayBuffer[Int] =
  ArrayBuffer(12, 15)
```

```
scala> buf.length
res17: Int = 2

scala> buf(0)
res18: Int = 12
```

2. Les Sets et Maps

```
scala> import scala.collection.mutable
import scala.collection.mutable
```

```
scala> val mutaSet = mutable.Set(1, 2, 3)
mutaSet: scala.collection.mutable.Set[Int] = Set(3, 1, 2)
```

```
scala> val text = "See Spot run. Run, Spot. Run!"
text: java.lang.String = See Spot run. Run, Spot. Run!

scala> val wordsArray = text.split("[ !,.]+")
wordsArray: Array[java.lang.String] =
   Array(See, Spot, run, Run, Spot, Run)
```

```
scala>  val words = mutable.Set.empty[String]
words: scala.collection.mutable.Set[String] = Set()
```

```
scala> for (word <- wordsArray)
     |   words += word.toLowerCase

scala> words
res25: scala.collection.mutable.Set[String] =
  Set(spot, run, see)
```

```
scala> val map = mutable.Map.empty[String, Int]
map: scala.collection.mutable.Map[String,Int] = Map()
```

```
scala> map("hello") = 1

scala> map("there") = 2

scala> map
res28: scala.collection.mutable.Map[String,Int] =
  Map(hello -> 1, there -> 2)
```

```
scala> map("hello")
res29: Int = 1
```

```
scala> import scala.collection.immutable.TreeSet
import scala.collection.immutable.TreeSet

scala> val ts = TreeSet(9, 3, 1, 8, 0, 2, 7, 4, 6, 5)
ts: scala.collection.immutable.SortedSet[Int] =
  Set(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

scala> val cs = TreeSet('f', 'u', 'n')
cs: scala.collection.immutable.SortedSet[Char] = Set(f, n, u)
```

```
scala> import scala.collection.immutable.TreeMap
import scala.collection.immutable.TreeMap

scala> var tm = TreeMap(3 -> 'x', 1 -> 'x', 4 -> 'x')
tm: scala.collection.immutable.SortedMap[Int,Char] =
  Map(1 -> x, 3 -> x, 4 -> x)

scala> tm += (2 -> 'x')

scala> tm
res38: scala.collection.immutable.SortedMap[Int,Char] =
  Map(1 -> x, 2 -> x, 3 -> x, 4 -> x)
```

3. Sélection de collections mutables ou immutables

```
scala> val people = Set("Nancy", "Jane")
people: scala.collection.immutable.Set[java.lang.String] =
  Set(Nancy, Jane)

scala> people += "Bob"
<console>:6: error: reassignment to val
```

```
       people += "Bob"
              ^
```

```
scala> var people = Set("Nancy", "Jane")
people: scala.collection.immutable.Set[java.lang.String] =
  Set(Nancy, Jane)

scala> people += "Bob"

scala> people
res42: scala.collection.immutable.Set[java.lang.String] =
  Set(Nancy, Jane, Bob)
```

```
scala> people -= "Jane"

scala> people ++= List("Tom", "Harry")

scala> people
res45: scala.collection.immutable.Set[java.lang.String] =
  Set(Nancy, Bob, Tom, Harry)
```

```
scala> var roughlyPi = 3.0
roughlyPi: Double = 3.0

scala> roughlyPi += 0.1

scala> roughlyPi += 0.04

scala> roughlyPi
res48: Double = 3.14
```

4. Initialisation des collections

```
scala> List(1, 2, 3)
res0: List[Int] = List(1, 2, 3)

scala> Set('a', 'b', 'c')
res1: scala.collection.immutable.Set[Char] = Set(a, b, c)

scala> import scala.collection.mutable
import scala.collection.mutable

scala> mutable.Map("hi" -> 2, "there" -> 5)
res2: scala.collection.mutable.Map[java.lang.String,Int] =
  Map(hi -> 2, there -> 5)

scala> Array(1.0, 2.0, 3.0)
res3: Array[Double] = Array(1.0, 2.0, 3.0)
```

```
scala> import scala.collection.mutable
import scala.collection.mutable

scala> val stuff = mutable.Set(42)
stuff: scala.collection.mutable.Set[Int] = Set(42)

scala> stuff += "abracadabra"
<console>:7: error: type mismatch;
 found   : java.lang.String("abracadabra")
 required: Int
       stuff += "abracadabra"
                ^
```

```
scala> val stuff = mutable.Set[Any](42)
stuff: scala.collection.mutable.Set[Any] = Set(42)

scala> stuff += "abracadabra"
res89: scala.collection.mutable.Set[Any] = Set(abracadabra, 42)
```

```
scala> val colors = List("blue", "yellow", "red", "green")
colors: List[java.lang.String] =
  List(blue, yellow, red, green)
```

```
scala> import scala.collection.immutable.TreeSet
import scala.collection.immutable.TreeSet

scala> val treeSet = TreeSet(colors)
<console>:6: error: no implicit argument matching
  parameter type (List[java.lang.String]) =>
    Ordered[List[java.lang.String]] was found.
       val treeSet = TreeSet(colors)
                     ^
```

```
scala> val treeSet = TreeSet[String]() ++ colors
treeSet: scala.collection.immutable.SortedSet[String] =
   Set(blue, green, red, yellow)
```

```
scala> treeSet.toList
res54: List[String] = List(blue, green, red, yellow)
```

```
scala> treeSet.toArray
res55: Array[String] = Array(blue, green, red, yellow)
```

```
scala> treeSet
```

```
  res5: scala.collection.immutable.SortedSet[String] =
    Set(blue, green, red, yellow)

  scala> val mutaSet = mutable.Set.empty ++ treeSet
  mutaSet: scala.collection.mutable.Set[String] =
    Set(yellow, blue, red, green)

  scala> val immutaSet = Set.empty ++ mutaSet
  immutaSet: scala.collection.immutable.Set[String] =
    Set(yellow, blue, red, green)
```

```
  scala> val muta = mutable.Map("i" -> 1, "ii" -> 2)
  muta: scala.collection.mutable.Map[java.lang.String,Int] =
     Map(ii -> 2, i -> 1)

  scala> val immu = Map.empty ++ muta
  immu: scala.collection.immutable.Map[java.lang.String,Int] =
     Map(ii -> 2, i -> 1)
```

5. Les Tuples

```
scala> val t = (4,3,2,1)
t: (Int, Int, Int, Int) = (4,3,2,1)

scala> val sum = t._1 + t._2 + t._3 + t._4
sum: Int = 10

scala> val(quatre,trois,deux,un)=t
quatre: Int = 4
trois: Int = 3
deux: Int = 2
un: Int = 1
```