Exemples de code pour la programmation en Scala

Table des matières

1. Apprendre à utiliser l'interprète Scala

```
sparkscala@sparkscala-VirtualBox:~$ scala
Welcome to Scala 2.12.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_181).
Type in expressions for evaluation. Or try :help.

scala>
```

```
scala> 1 + 2
res0: Int = 3
```

```
scala> res0 * 3
res1: Int = 9
```

```
scala> println("Hello, world!")
Hello, world!
```

2. Les opérations

Arithmétique

```
scala> 1.2 + 2.3
res1: Double = 3.5

scala> 3 - 1
res2: Int = 2

scala> 11 / 4
res3: Int = 2

scala> 11 % 4
res4: Int = 3

scala> 11.0f / 4.0f
res5: Float = 2.75

scala> 11.0 % 4.0
res6: Double = 3.0
```

Relationnel et logique

```
scala> 1 > 2
res16: Boolean = false

scala> 1 < 2
res17: Boolean = true
```

```
scala> 1.0 <= 1.0
res18: Boolean = true

scala> 3.5f >= 3.6f
res19: Boolean = false

scala> val thisIsBoring = !true
thisIsBoring: Boolean = false

scala> !thisIsBoring
res21: Boolean = true
```

```
scala> val toBe = true
toBe: Boolean = true

scala> val question = toBe || !toBe
question: Boolean = true

scala> val paradox = toBe && !toBe
paradox: Boolean = false
```

```
scala> def salt() = { println("salt"); false }
salt: ()Boolean

scala> def pepper() = { println("pepper"); true }
pepper: ()Boolean

scala> pepper() && salt()
pepper
salt
res22: Boolean = false

scala> salt() && pepper()
salt
res23: Boolean = false
```

Égalité d'objet

```
scala> 1 == 2
res31: Boolean = false

scala> 1 != 2
res32: Boolean = true

scala> 2 == 2
res33: Boolean = true
```

```
scala> List(1, 2, 3) == List(1, 2, 3)
res34: Boolean = true

scala> List(1, 2, 3) == List(4, 5, 6)
res35: Boolean = false
```

```
scala> 1 == 1.0
res36: Boolean = true

scala> List(1, 2, 3) == "hello"
res37: Boolean = false
```

```
scala> List(1, 2, 3) == null
res38: Boolean = false

scala> null == List(1, 2, 3)
res39: Boolean = false
```

```
scala> ("he"+"llo") == "hello"
res40: Boolean = true
```

3. Définir des variables

```
scala> val msg = "Hello, world!"
msg: java.lang.String = Hello, world!
```

```
scala> val msg2: java.lang.String = "Hello again, world!"
msg2: java.lang.String = Hello again, world!
```

```
scala> val msg3: String = "Hello yet again, world!"
msg3: String = Hello yet again, world!
```

```
scala> println(msg)
Hello, world!
```

```
scala> msg = "Goodbye cruel world!"
<console>:5: error: reassignment to val
       msg = "Goodbye cruel world!"
             ^
```

```
scala> var greeting = "Hello, world!"
greeting: java.lang.String = Hello, world!
```

```
scala> greeting = "Leave me alone, world!"
greeting: java.lang.String = Leave me alone, world!
```

```
scala> val multiLine =
     |   "This is the next line."
multiLine: java.lang.String = This is the next line.
```

```
scala> val oops =
      |
      |
You typed two blank lines.  Starting a new command.
```

4. Définir certaines fonctions

```
scala> def max(x: Int, y: Int): Int = {
     |   if (x > y) x
     |   else y
     | }
max: (Int,Int)Int
```

```
scala> def max2(x: Int, y: Int) = if (x > y) x else y
max2: (Int,Int)Int
```

```
scala> max(3, 5)
res6: Int = 5
```

```
scala> def greet() = println("Hello, world!")
greet: ()Unit

scala> greet()
Hello, world!
```

```
scala> :quit
```

5. Ecrire des scripts Scala

Créer le script **hello.scala** avec le code :

```
println("Hello, world, from a script!")
```

Faire un run avec la commande :

```
$ scala hello.scala
```

Résultat

```
Hello, world, from a script!
```

```
Nom script :
  helloarg.scala

Code :
  // Say hello to the first argument
  println("Hello, "+ args(0) +"!")

 Run :
   $ scala helloarg.scala planet

Résultat :
  Hello, planet!
```

6. Structure de contrôle

If / Else

```
if ( 12 < 7) println("Etrange") else println("C'est mieux :-)")

if ( 12 < 7) {
     println("Etrange")
   } else {
     println("C'est mieux :-)")
   }
```

Match

```
def matchTest(x: Int): String = x match {
  case 1 => "un"
  case 2 => "deux"
  case _ => "autre chose"
}

matchTest(3)  // many
matchTest(1)  // one
```

7. Boucle while

```
Nom script :
  printargs.scala

Code :
  var i = 0
  while (i < args.length) {
    println(args(i))
    i += 1
```

```
  }

Run :
  $ scala printargs.scala Scala est bien

Résultat :
  Scala
  est
  bien
```

```
Nom script :
  echoargs.scala

Code :
  var i = 0
  while (i < args.length) {
    if (i != 0)
      print(" ")
    print(args(i))
    i += 1
  }
  println()

Run :
  $ scala echoargs.scala Scala is even more fun

Résultat :
  Scala is even more fun
```

---

8. Itérer avec foreach et for

```
Nom script :
  pa.scala

Code :
  args.foreach(arg => println(arg))

Run :
  $ scala pa.scala Concise is nice

Résultat :
  Concise
  is
  nice
```

```
Nom script :
  forargs.scala

Code1 :   args.foreach((arg: String) => println(arg))
Code2 :   args.foreach(println)
Code3 :   for (arg <- args) println(arg)

Run :
```

```
$ scala forargs.scala for arg in args
```

Résultat :

```
for
arg
in
args
```