Big Data Technologies

2024/2025

Lab 1 - SCALA

Starting procedure

- Build the container
  `docker build -t scalaapp:v1 .`
- Run it with the command
  `docker run -it -p 8888:8888 scalaapp:v1 /usr/bin/bash`
- Open a browser on your laptop and go to
  `http://127.0.0.1:8888/lab?token=datatoken`
- Open a scala notebook
- Do the following exercises in a single notebook
- At the end of lab, go into your container via the terminal/command prompt, Ctrl+C to close jupyterlab. You will leave automatically the container.

Note: if your container does not work, you can use https://scastie.scala-lang.org for this lab.

PART 1: first commands

- Try the following commands:

```
> 2 + 5 * 3
> "Hello, " + "World !"
> "Hello, ".concat("World !")
> "Hello"(0)
>  true || false
> 2.2 < 2
> 5 == -(-5)
> "Bonjour" == "Good Morning"
> if (0 > 1) "Bonjour" else "Bonsoir"
> val x: java.lang.String = "A String !"
> var a: Int = if (5 > 6) 8 else 9
```

- Compare:

```
> var pi: Double = 3.141592653
> pi += 2
```

and

```
> val pi: Double = 3.141592653
> pi += 2
```

- Compare:

```
> (12 % 3) * (5 / 2)
```

and

```
> (12 / 3) * (5 / 2)
```

- Compare

> val a = 2 ; println(a * 3)

and

> val a = 2 ; println(a.toString * 3)

- Does the following command work? Why?

> val A=List("red","blue","white")
> A(0)="green"

- Does the following command work? Why?

> val B=Array("red","blue","white")
> B(0)="green"

PART 2: first class

Create the class “student” with the following code:

```
class student {
      var name = "myName"
      var age = 0
      def increaseAge : Unit = { age = age + 1 }
   }
```

1- Create a method called “printStudent” which prints the name and the age of the student.
2- Create a list called “listGrades” which can store the grades of student (a grade is a Double). We can add a grade (not remove it) by using a method called “addGrade”.
3- Create a method called “meanGrade” which computes the mean of the student.

PART 3: first program

Create a scala object “wordCount” containing a main function called “main” whose argument is a String called “sentence”.
This object will call a class called “wordCountClass” which stores a string and has a method called “counts” counting the word occurrences within the string.
The program must count the word occurrences within the string “sentence” by using an object of “wordCountClass”.

For example, if the user enters the sentence “one two one two three four one”, the output is
(one,3)
(three,1)
(four,1)
(two,2)

Hints: some Scala methods like “foreach”, “split”, “groupBy” and “map” could be useful. The function “map” is described in the lecture.

PART 4: first functions

1- Create a function "fSum" which takes two integers x and y as inputs and compute the cumulative sum from x to y (including x and y). For example, calling fSum(3,7) gives 25 and fSum(5,18) gives 161.

2- Create an other function "fSumTwice" which takes fSum and two integers x, y as inputs and whose result is fSum(0,fSum(x,y)). For example, fSumTwice(fSum,3,5) gives 78.

PART 5: pattern matching

Write a program with the following code:

```
abstract class Expression
case object X extends Expression
case class Const(value: Int) extends Expression
case class Add(left: Expression, right: Expression) extends Expression
case class Mult(left: Expression, right: Expression) extends Expression
case class Neg(expr: Expression) extends Expression

def eval(expression: Expression, xValue: Int): Int = expression match {
  to complete…
}

println(eval(X,3)) // result: Int = 3
println(eval(Const(2),3)) // result: Int = 2
println(eval(Add(X,X),3)) // result: Int = 6
println(eval(Add(X,Const(2)),3)) // result: Int = 5
println(eval(Neg(X),3)) // result: Int = -3

val expr = Neg(Add(Const(1), Mult(Const(2), Mult(X, X)))) // -(1 + 2 * X*X)
println(eval(expr, 3)) // result: Int = -19
```

You must complete the function "eval" by using pattern matching. The function "eval" should be able to process all kinds of case classes. The commented results of "eval" given in the code should help you to understand how "eval" works.