

# Kojo/Scala



Andrea Sterbini – [sterbini@di.uniroma1.it](mailto:sterbini@di.uniroma1.it)

# Kojo learning environment: Scala language + turtles + math/geometry

## Kojo IDE:

- HTML “stories” to guide you
  - Code editor
  - 2D canvas (turtle-based)
  - ~~- 2D math and geometry cartesian canvas (Geogebra): - [OLD]~~
    - ~~- lines, points, segments, angles, areas, formulas, plots, ...~~
  - Arithmetic exercises
  - Music player (with Midi instruments)
  - Game programming (Processing-like Stage) [NEW]
- Other: Arduino programming (with an arduino driver/interface)

# Scala programming language

## Scala functional programming language:

- compiles to the Java VM (and can import Java libs)
- simpler readable syntax (e.g. object method parameter )
- functions + OOP
  - anonymous functions, map/filter/...
  - simplified definition of classes (no need for getters/setters)
- preferred immutable structures vs. mutable:
  - immutable (**val**) vs. mutable (**var**) variables
  - imm. collections are copied while mutable can be changed
  - method arguments are always immutable
- operator overloading
- recursive structure matching (“similar” to unification)
  - multiple assignment (like Python)

# Scala resources

## Programming in Scala

[by Martin Odersky, Lex Spoon, and Bill Venners]

<https://www.scala-lang.org>

the Scala language community

## Good IDE for Scala:

- Eclipse
- Netbeans

# Programming style

Single-threaded? YES (Concurrency? YES )

Procedural? YES (Lazy evaluation? YES )

**Functional? YES!**

Statically typed:

- complex abstract types
- metaprogramming

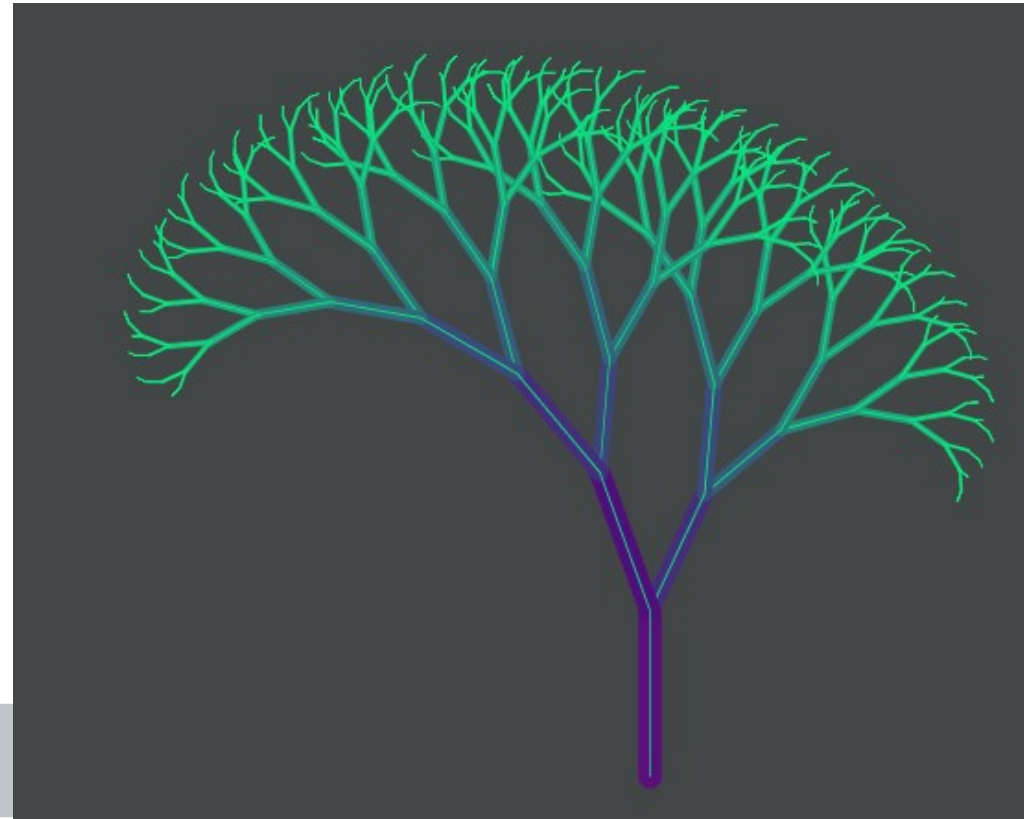
Data types:

- Objects + Classes (+ Singletons + Structs)
- sequential/parallel collections
- list-based operations (map/filter/...)
- anonymous functions (code blocks)

# Demo 1: recursive tree by turtle graphics (as usual)

```
def tree(distance: Double) {  
  if (distance > 4) {  
    setPenThickness(distance/7)  
    setPenColor(  
      Color(distance.toInt, math.abs(255-distance*3).toInt, 125))  
    forward(distance)  
    right(25)  
    tree(distance*0.8-2)  
    left(45)  
    tree(distance-10)  
    right(20)  
    hop(-distance)  
  }  
}
```

```
clear()  
invisible()  
setAnimationDelay(10)  
hop(-200)  
tree(90)
```



# Good debugger with recursion trace/visualization

The Kojo Learning Environment

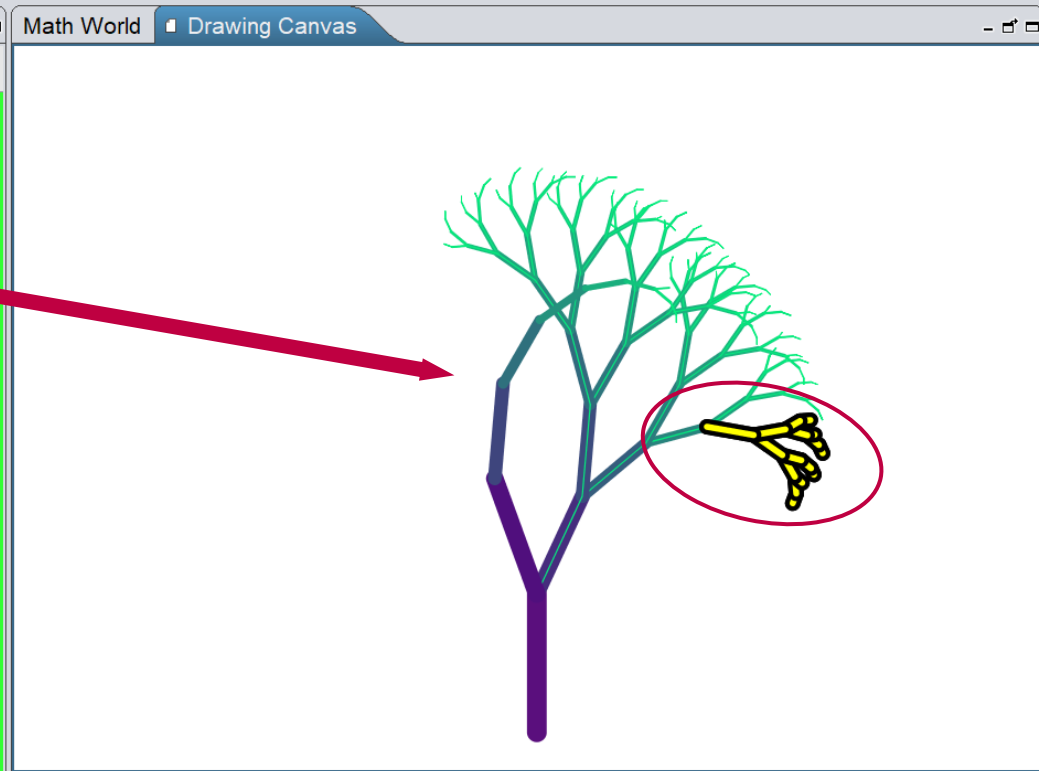
File Samples Showcase Window Language Tools Help

Program Trace x

- CALL setPenThickness (t
- CALL abs (x = 131.399999
- RETURN abs = 131.399999
- CALL Color (r = 41, g =
- RETURN Color = java.awt
- CALL setPenColor (color
- CALL forward (distance = 41.2)
- CALL right (angle = 25.0
- CALL tree (distance = 30
- CALL setPenThickness
- CALL abs (x = 162.12)

Name: tree  
Args: (distance = 30.96)  
Call Level: 5  
Target Object: Wrapper\$UserCode\$@171e916  
Target Type: Wrapper\$UserCode\$  
Source: scripteditor  
Entry Line Number: 2  
Exit Line Number: 0  
Caller Source: scripteditor  
Caller Line Number: 7  
Source Line: if (distance > 4) {  
Caller Source Line: tree(distance\*0.8-2)

```
1 def tree(distance: Double) {  
2   if (distance > 4) {  
3     setPenThickness(distan  
4     setPenColor(Color(dist  
5     forward(distance)  
6     right(25)  
7     tree(distance*0.8-2)  
8     left(45)  
9     tree(distance-10)  
10    right(20)  
11    back(distance)  
12  }  
13 }  
14  
15 clear()  
16 invisible()  
17 setAnimationDelay(10)  
18 penUp()  
19 back(200)  
20 penDown()  
21 tree(90)
```



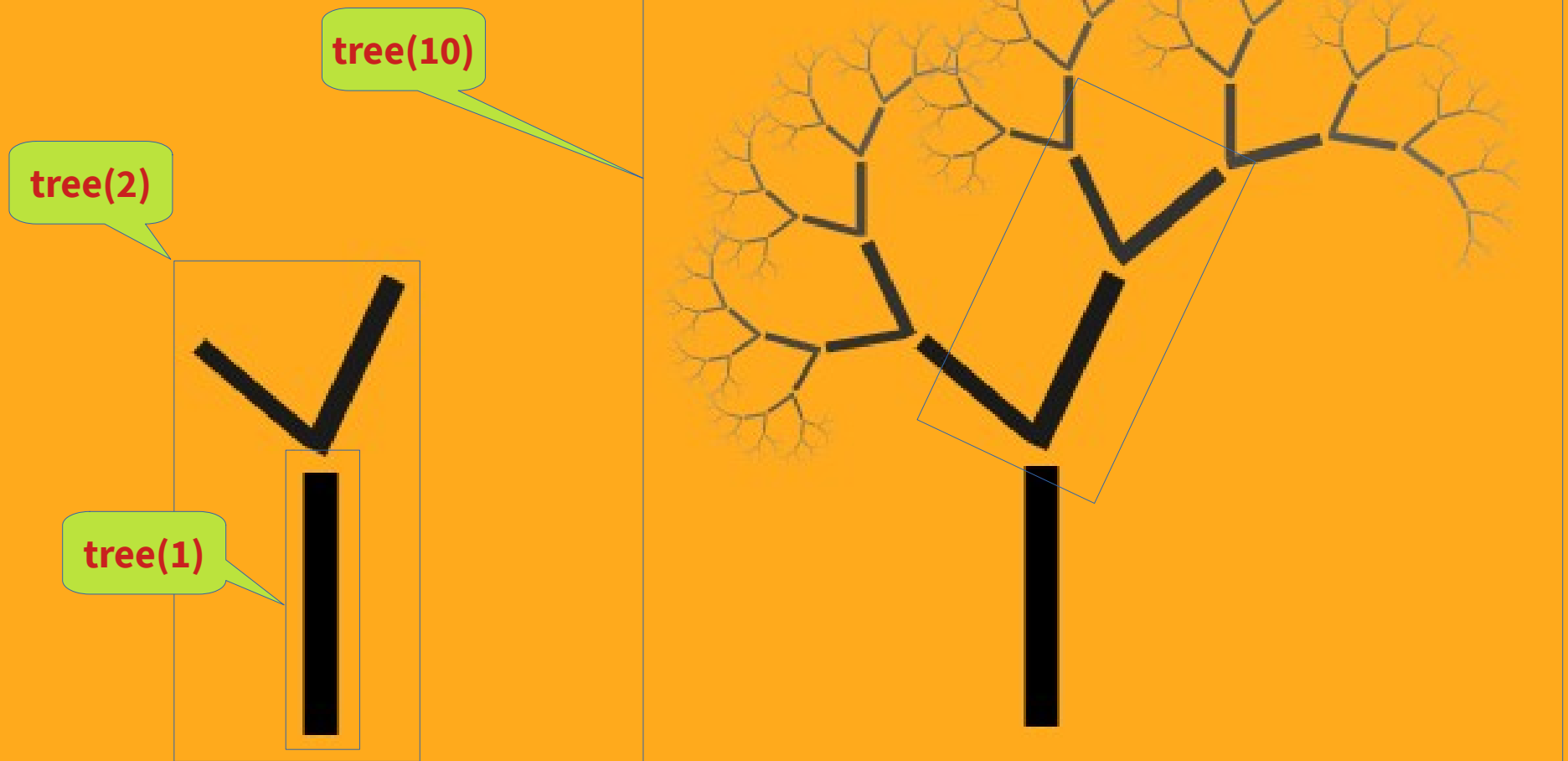
Output Pane

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

Mouse Position: (-132, -59)

8 | 1

**Tree2 = recursive transformations of a rectangle  
rotate(-25°) + scale(72%) + brightness(+10%)**





# Demo 2: recursive tree = recursive pictures + graphic transformations

```
// we start with a square
def square = Picture {
  repeat (4) {
    forward(100)
    right()    // default 90°
  }
}
```

```
// a stem is a distorted black square
def stem = scale(0.13, 1) *
  penColor(noColor) *
  fillColor(black) -> square
```

```
def drawing(n: Int): Picture = {
  if (n < 2)
    stem
  else
    GPics(stem,
      trans(0, size + 10) * brit(0.1) -> Gpics(
        rot(-25) * scale(0.72) -> drawing(n-1),
        rot( 50) * scale(0.55) -> drawing(n-1)
      )
    )
}

clear()
setBackground(Color(255, 170, 29))
invisible()
val pic = trans(0, -300) -> drawing(10)
draw(pic)
```

## Demo 3

[S. Penge]

### choose the correct article for an italian word

Type: definite/indefinite (determinativo/indeterminativo)

Gender: male/female

Number: singular/plural

1) deduce the word gender from final char

2) select proper gender/number from final char

3) handle normality and exceptions (here for ind. male sing. only)

- starts with vowel → "un"
- starts with consonant → "un"
- starts with 2 special vowels ('ia', 'ie', 'io', 'iu') → "uno"
- starts with 1 or 2 special consonants → "uno"  
( "x", "y", "z", "gn", "pt", "ps", "pn", "sc", "sf", "sq", "st")

# Demo 4

[S. Penge]

## Music transposition and art

- Many voices/instruments
- Transposed to a different key/mode (major/minor)
- Each voice produces a coloured path (each note a segment)
- With arrangement



## ~~Demo 5: MathWorld pane = Geogebra [V 2.7]~~

~~Just use turtle graphics with the Mw class to create lines and angles~~

```
val t = Mw.turtle(-1, 0)
```

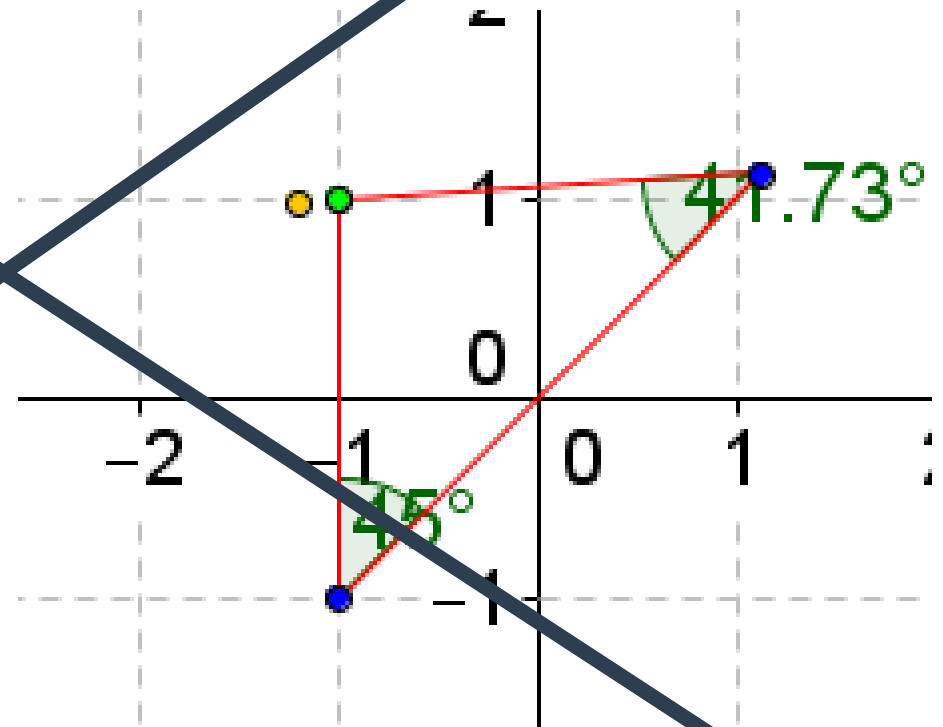
```
t.showExternalAngles()
```

```
t.forward(-2)
```

```
t.right(45)
```

```
t.forward(3)
```

```
t.moveTo(-1,0)
```



~~Full GeoGebra with spreadsheet~~

~~Not (yet?) possible to get all properties FROM Geogebra elements~~

# NEW

## Game programming with “Staging”

[V 2.9]

```
import Staging._ // Import the Staging library
import Staging.{circle, clear, animate} // explicitly import names that clash
clear()
gridOn()
rectangle(0,0,200,100)
val ball = circle(-200, -100, 5)
var y = 0 ; var x = 0 // ball position
var dy = 10; var dx = 3 // ball speed
Animate { // animation is around 30 - 50 frames per second
    ball.setPosition(x,y)
    // update ball position, detect end of bounce area
    dx = if(x < 0 || x > 200) -dx else dx
    x += dx
    dy = if(y < 0 || y > 100) -dy else dy
    y += dy
}
```

# Arithmetic exercises

## Sum, Multiplication, Division, Subtraction

Numbers to be Added:

Digits per Number:

Difficulty Level:

Time Limit:

	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	7	7	8	9
	4	5	8	6
+	6	4	4	5

Digits per Number:

Difficulty Level:

Time Limit:

	<input type="text"/>	<input type="text" value="6"/>	<input type="text"/>	<input type="text"/>
	3	<del>7</del>	12	5
-	2	4	4	5

Congratulations! You've got it right. Your grade (based on time taken and mistakes made) is:

A

Digits in Dividend:

Difficulty Level:

Time Limit:

	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
	3	4	4	7	0

Digits per Number:

Difficulty Level:

Time Limit:

	<input type="text"/>	<input type="text" value="6"/>	<input type="text"/>	<input type="text"/>
	3	<del>7</del>	12	5
-	2	4	4	5

Congratulations! You've got it right. Your grade (based on time taken and mistakes made) is:

A

**Demo**

**DEMO**