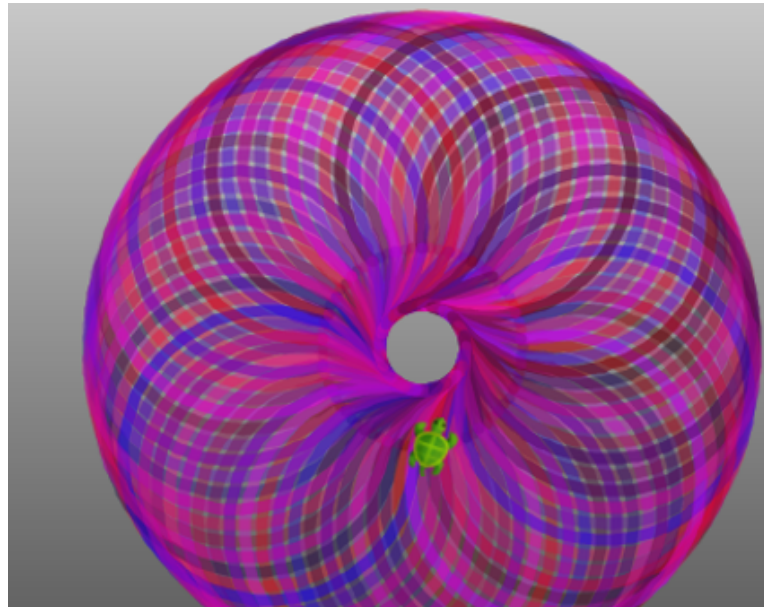


# Challenges with Kojo

Editor: Björn Regnell  
[www.lth.se/code](http://www.lth.se/code)



# Challenges with Kojo

Version: April 8, 2015



License: Creative Commons *Attribution-NonCommercial-ShareAlike 4.0 International* CC BY-NC-SA 4.0

Editor: Björn Regnell

Translation to English: Björn Regnell, Lalit Pant, Fatima Abou Alpha, Rasha El-Manzalawy, Love Sjögren

Contributors: Björn Regnell, Lalit Pant, Sandra Nilsson, Maja Johansson, Simone Strippgen, Christoph Knabe

© Björn Regnell, Lund University, 2015

<http://lth.se/programera>

# Contents

About Kojo	1	Draw several polygons	16	Save the animals in a vector	34
Your first program	2	Values and expressions	17	Practice words	35
Draw a square	3	Name values with <code>val</code>	18	Capital Game	36
Draw stairs	4	Random numbers	19	Make a timer with <code>object</code>	37
Make a loop	5	Mix your own colors	20	Simulate a traffic light	38
Draw a character	6	Try the color chooser	21	Control the turtle with the keyboard	39
How fast is your computer?	7	Draw random circles	22	Control the turtle with the mouse	40
Trace the program	8	Draw a flower	23	Make your own bank account	41
Write your own function with <code>def</code>	9	Create a variable with <code>var</code>	24	Make many objects from a <code>class</code>	42
Stack squares	10	Draw many flowers	25	Talk to the computer	43
Make a stack-function	11	Change the turtle's costume	26	Modify the pong game	44
Make a grid	12	Make many turtles with <code>new</code>	27		
A square with parameter	13	Make a turtle race	28		
Draw a square character	14	Alternative with <code>if</code>	29		
Draw a polygon	15	React to what the user is doing	30		
		Make a <code>while</code> -loop	31		
		Guess the number	32		
		Practice multiplication	33		

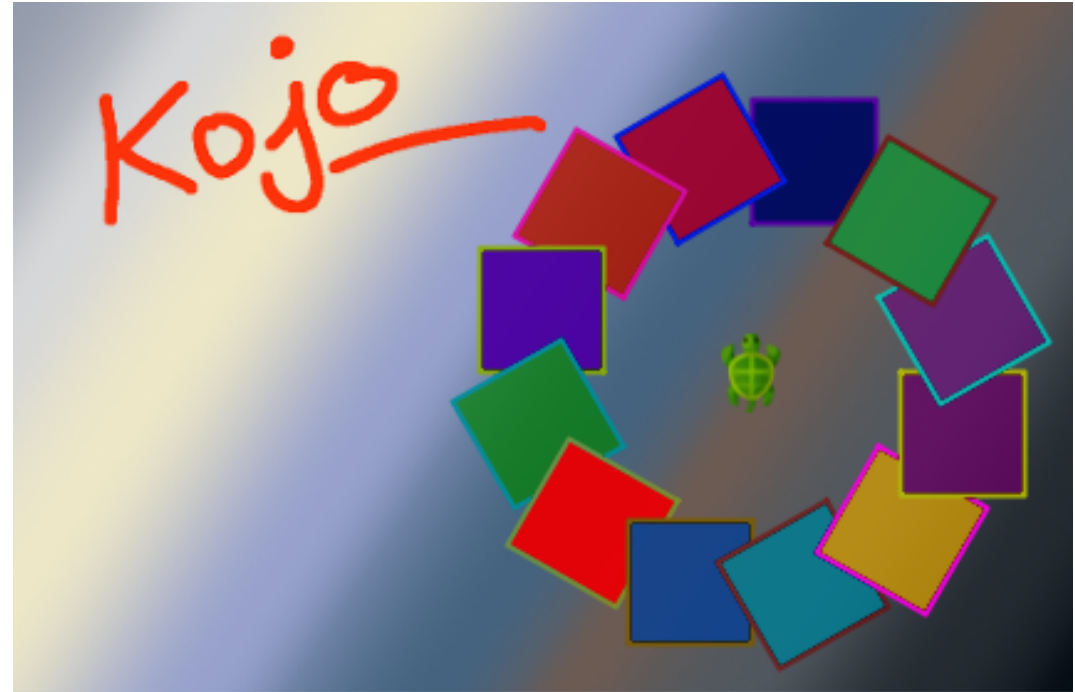
# About Kojo

## What is Kojo?

Kojo is an app that can help you learn how to program. With Kojo you can code using the modern and powerful programming language **Scala**. Kojo is free and available for Linux, Windows and Mac.

## Where can I find Kojo?

Download Kojo here:  
[www.kogics.net/kojo-download](http://www.kogics.net/kojo-download)  
Read more here:  
[www.kogics.net/kojo](http://www.kogics.net/kojo)



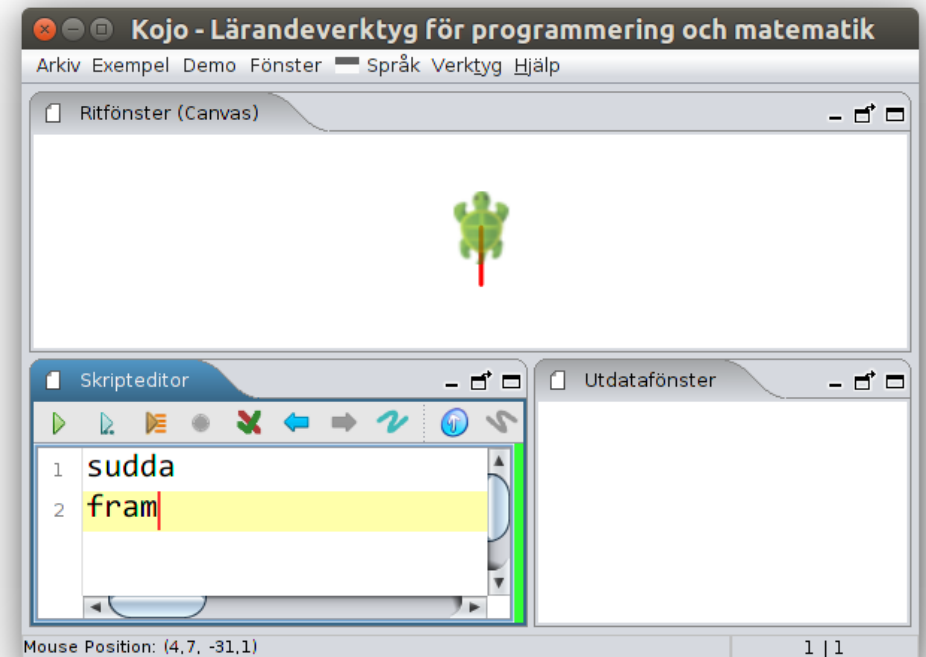
# Your first program

## Challenge:

Write the following in the Kojo script editor window:

```
clear  
forward
```

Press the green play button  
to run your program.



# Draw a square

```
clear  
forward  
right
```

If you write left or right the turtle will change direction.

## Challenge:

Extend the program so that it makes a square.



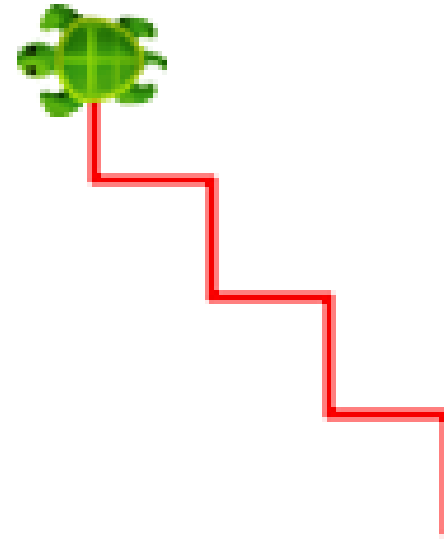
# Draw stairs

```
clear  
forward; left  
forward; right
```

With semicolon ; between the commands, you could have several commands on the same line.

## Challenge:

Extend the program so that it makes stairs.



# Make a loop

```
clear  
repeat(4){ forward; right }
```



## Challenge:

- What will happen if you change 4 to 100?
- Draw stairs with 100 steps.



# Draw a character

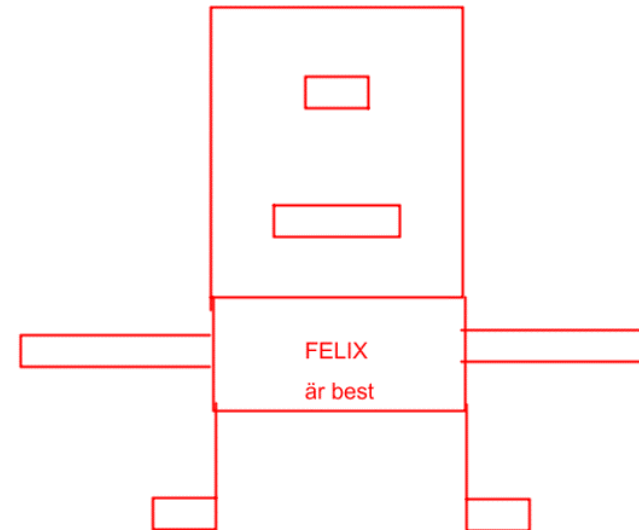
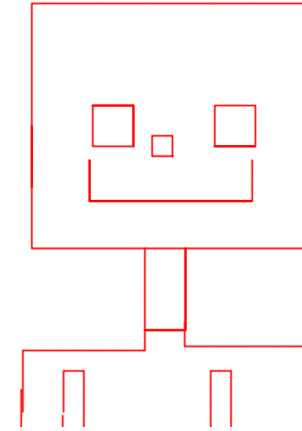
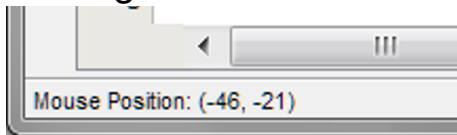
## Challenge:

Draw a character of your choice.

## Tip:

```
hop  
left(180)  
forward(300)  
hop(100)  
jumpTo(25,-28)  
write("FELIX is awesome")  
setPenColor(purple)  
setFillColor(green)
```

You can see the turtle's position down to the left while moving the mouse in the Canvas:



# How fast is your computer?

The first electronic computer was called **ENIAC** and could count up to 5000 in a second. In Kojo there is a function `räknaTill` that measures how fast the computer counts. When I run `räknaTill(5000)` on my fast computer, the following appears in the output window:

```
*** Räknar från 1 till ... 5000 *** KLAR!  
Det tog 0.32 millisekunder.
```

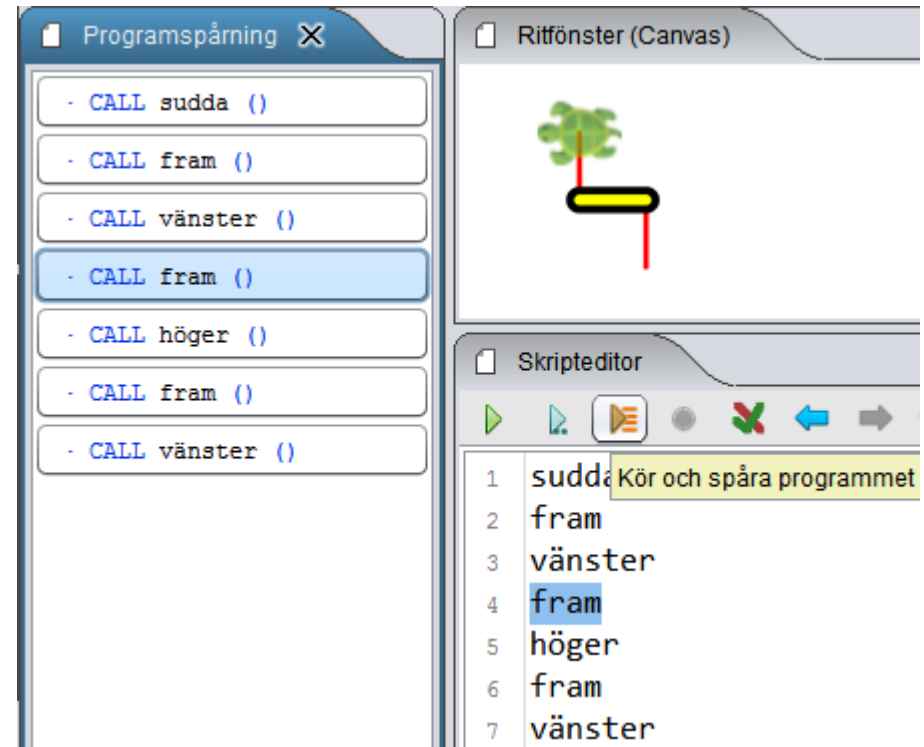
## Challenge:

- Run `räknaTill(5000)` and check if your computer is faster than mine.
- How long does it take for your computer to count up to a million?
- How much can your computer count to in a second?

# Trace the program

## Challenge:

- Write a program that draws stairs.
- Press the orange play button.
- Press on one of the commands: CALL fram. What happens in the Canvas?
- When a part of the program is marked in blue, only that part will run when you press the play button. You can unmark the code if you click next to the code that is marked.
- Add more commands to your program and observe what happens when you track it.
- Close the *Program trace* window when you are done.



# Write your own function with **def**

With **def** you can write your own *functions* and choose their names.

```
def square = repeat(4){ forward; right }
```

```
clear
```

```
square    //use your square-function
```

```
hop
```

```
square
```

## Challenge:

- Change the color of the squares.
- Make several squares.

## Tip:

```
setFillColor(green); setPenColor(purple)
```

# Stack squares

## Challenge:

Make a stack of 10 squares.

## Tip:

```
def square = repeat(4){ forward; right }
```

```
clear; setAnimationDelay(100)  
repeat(10){ ??? }
```



# Make a stack-function

## Challenge:

Make a function called `stack`, that draws a stack of 10 squares.

## Tip:

```
def square = repeat(4){ forward; right }  
def stack = ???
```

```
clear; setAnimationDelay(100)  
stack
```



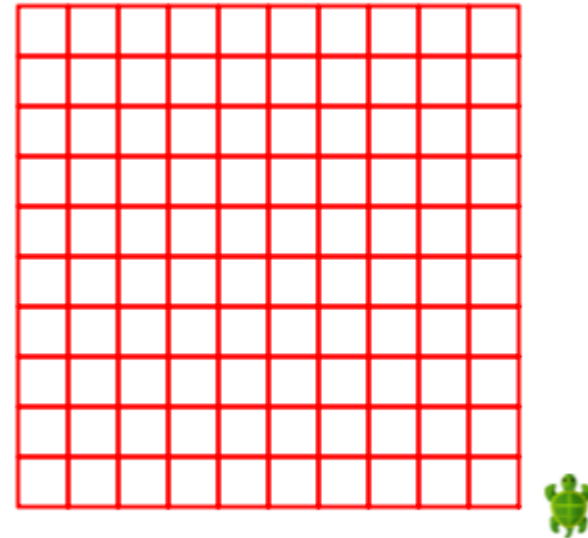
# Make a grid

## Challenge:

Make a grid of 10\*10 squares.

## Tip:

- Use your stack-function that you created earlier.
- You can jump backwards a whole column with `hop(-10 * 25)`
- You can then jump to the right position with `right; hop; left`



# A square with parameter

## Challenge:

Draw squares in different sizes.

## Tip:

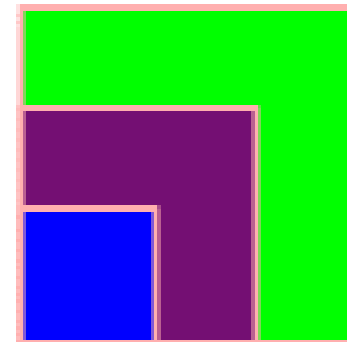
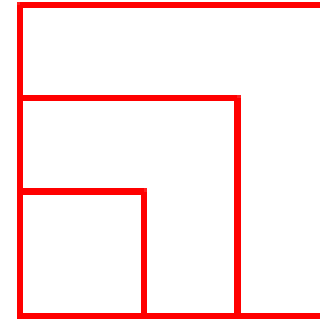
Give your function a *parameter*, called side of the type Int:

```
def square(side : Int) =  
  repeat(4){ forward(side); right }
```

```
clear; setAnimationDelay(100); invisible  
square(100)  
square(70)  
square(40)
```

You can change the color with:

```
setFillColor(blue); setPenColor(pink)
```





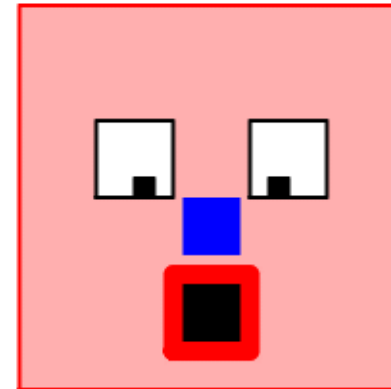
# Draw a square character

## Challenge:

Draw a character with squares of different sizes.

## Tip:

```
def square(x: Int, y: Int, side: Int) = {  
  jumpTo(x, y)  
  repeat(4) { forward(side); right }  
}  
def head(x: Int, y: Int) = { setFillColor(pink); setPenColor(red); square(x, y, 200) }  
def eye(x: Int, y: Int) = { setFillColor(white); setPenColor(black); square(x, y, 40) }  
def pupil(x: Int, y: Int) = { setFillColor(black); setPenColor(black); square(x, y, 10) }  
def nose(x: Int, y: Int) = { setFillColor(blue); setPenColor(noColor); square(x, y, 30) }  
def mouth(x: Int, y: Int) = { setPenThickness(10); setFillColor(black); setPenColor(red); square(x, y, 40) }  
  
clear; setAnimationDelay(20); invisible  
head(0, 0)  
eye(40, 100); pupil(60, 100)  
???
```



# Draw a polygon

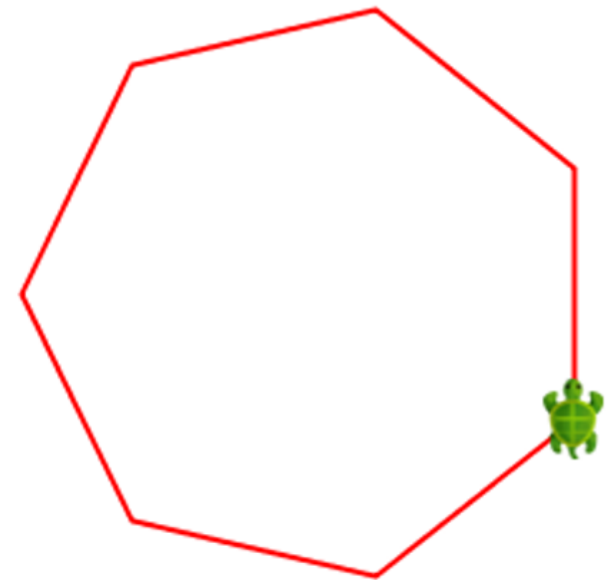
## Challenge:

- Try out the code below. Draw different kinds of polygons.
- Add a parameter `side` and draw polygons of different sizes.
- How large does `n` have to be to make it look like a circle?

## Tip:

```
def polygon(n:Int) = repeat(n){  
  forward(100)  
  left(360.0/n)  
}
```

```
clear; setAnimationDelay(100)  
polygon(7)
```

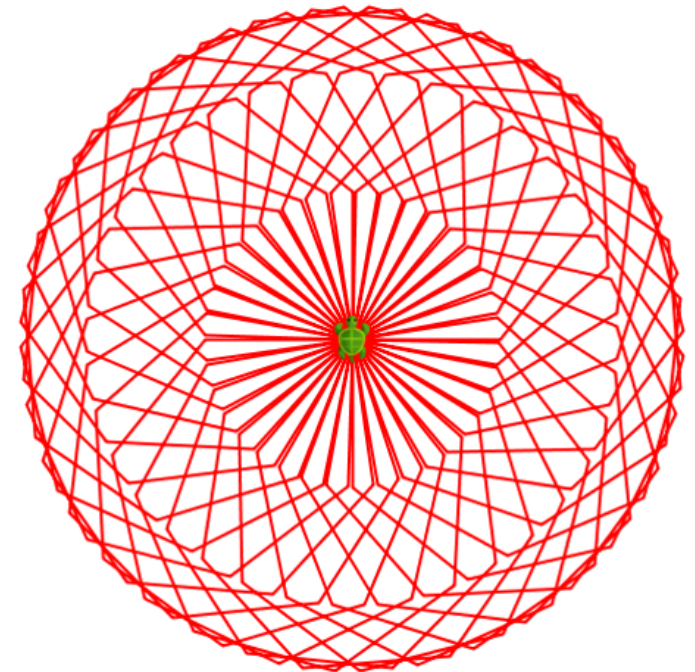


# Draw several polygons

## Challenge:

- Try out the program below.
- Try to change the amount of sides and the angle.
- Fill the polygons with different colors.

```
def polygon(n: Int, side: Int) = repeat(n){  
  forward(side)  
  left(360.0/n)  
}  
def rotate(n: Int, heading: Int, side: Int) =  
  repeat(360/heading){ polygon(n, side); left(heading) }  
  
clear; setAnimationDelay(5)  
rotate(7, 10, 100)
```

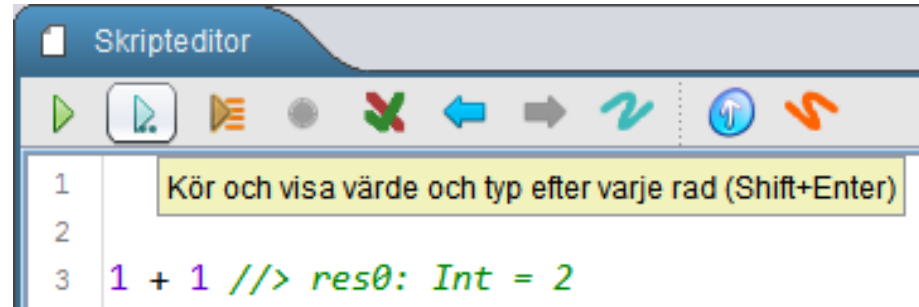


# Values and expressions

## Challenge:

- Write `1 + 1` and press the blue play button. Kojo will then create a green comment.
- The comment shows the value of the expression `1 + 1` that is 2 and the type is `Int`, which means integer.
- Create more expressions. What are the values and types of the expressions below?

```
5 * 5
10 + 2 * 5
"Hello" + "world"
5 / 2
5 / 2.0
5 % 2
```



## Tip:

- `/` between integers values results in integer division ignoring the decimals. To make a division with decimals, make sure at least one of the numbers have decimals. The type of a decimal value is called `Double`.
- With `%` you get the remainder of a division of integers.

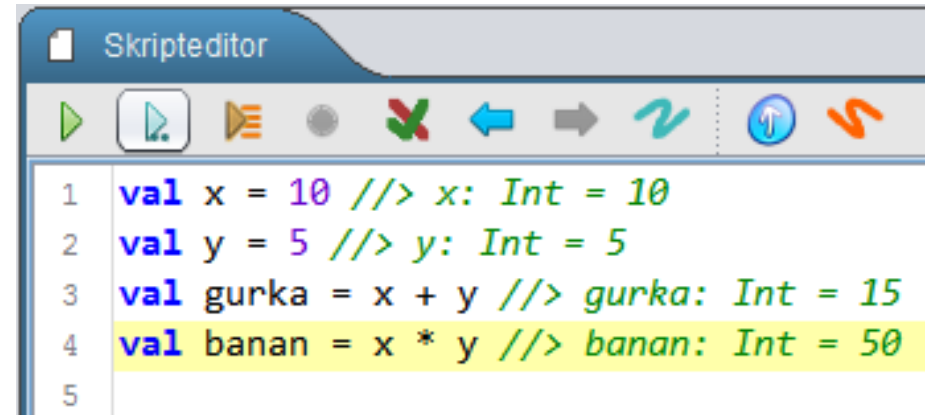
# Name values with **val**

## Challenge:

With **val** you can connect a name to a value. The name can then be used instead of the value. Try out the program below. What does the turtle write?

```
val x = 10
val y = 5
val cucumber = x + y
val banana = x * y
```

```
clear
forward; write(banana)
forward; write(cucumber)
forward; write(y)
forward; write(x)
```



```
Skripteditor
1 val x = 10 //> x: Int = 10
2 val y = 5 //> y: Int = 5
3 val gurka = x + y //> gurka: Int = 15
4 val banan = x * y //> banan: Int = 50
5
```

# Random numbers

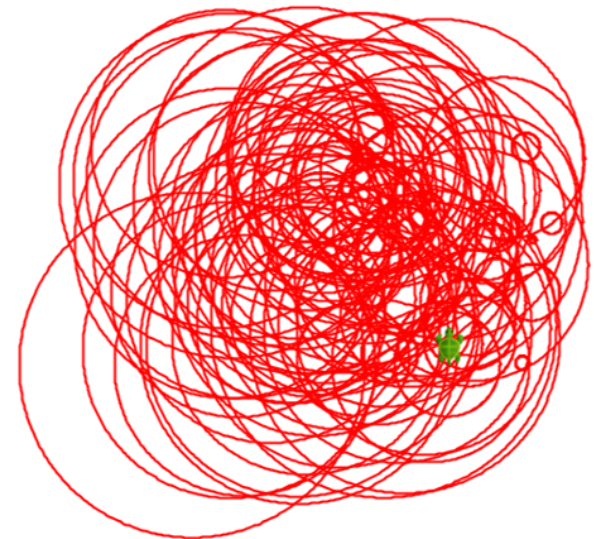
## Challenge:

- Run the program below several times. What happens?
- What is the smallest and largest possible value of the radius  $r$ ?
- Change it so that  $r$  becomes a random number between 3 and 200.
- Draw 100 circles, each with a random radius at a random position, as shown in the picture.

//r becomes a random number between 10 and 89:

```
val r = random(90) + 10
```

```
clear; setAnimationDelay(10); invisible  
write("Radius = " + r)  
circle(r)
```



# Mix your own colors

- You can mix your own colors with `Color`, for example `Color(0, 70, 0)`
- The three parameters are the values for *red*, *green* och *blue*
- You are also able to add a fourth parameter that sets the *transparency*
- All parameters are between 0 and 255

## Challenge:

Try the program below. Change the transparency

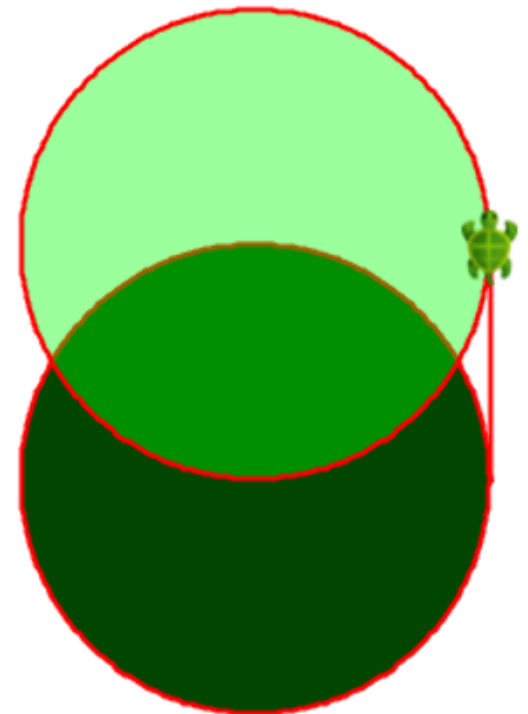
```
clear; setAnimationDelay(100)
```

```
val olivegreen = Color(0,70,0)
```

```
val pistageicecream = Color(0,255,0,100)
```

```
setFillColor(olivegreen); circle(100)
```

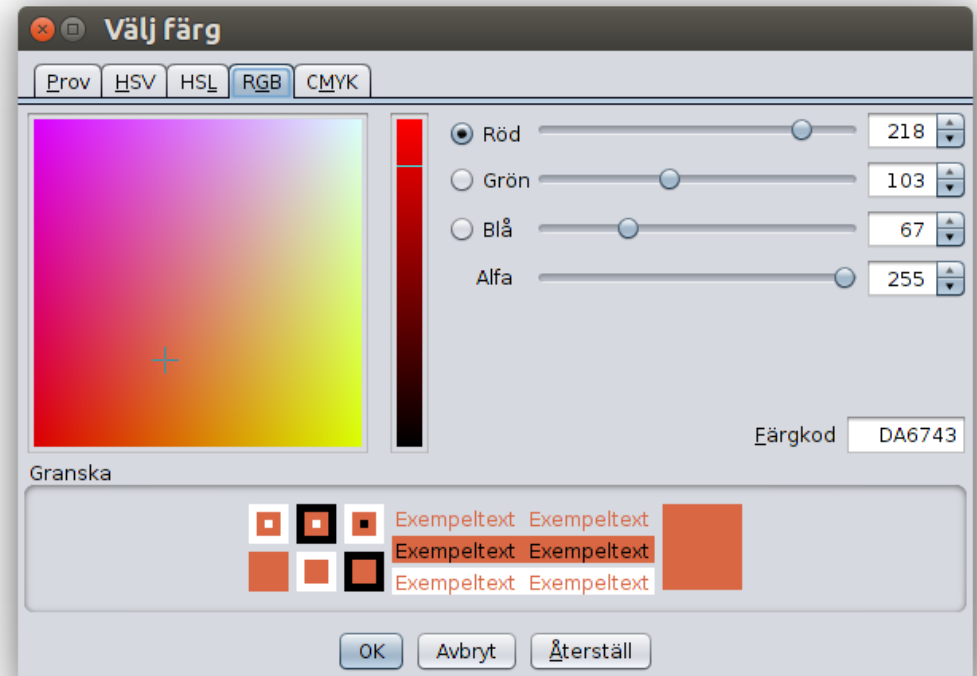
```
setFillColor(pistageicecream); forward(100); circle(100)
```



# Try the color chooser

## Challenge:

- Right-click in the editorwindow and click Choose color...
- If you choose the tab **RGB** in the color picker you can pick amongst new RGB-colors.
- Press OK and look in the output twindow. There you can see the three RGB-values for red, green and blue.
- You can use these values in your program to draw your new color with `color(Color(218,153,67))`.





# Draw random circles

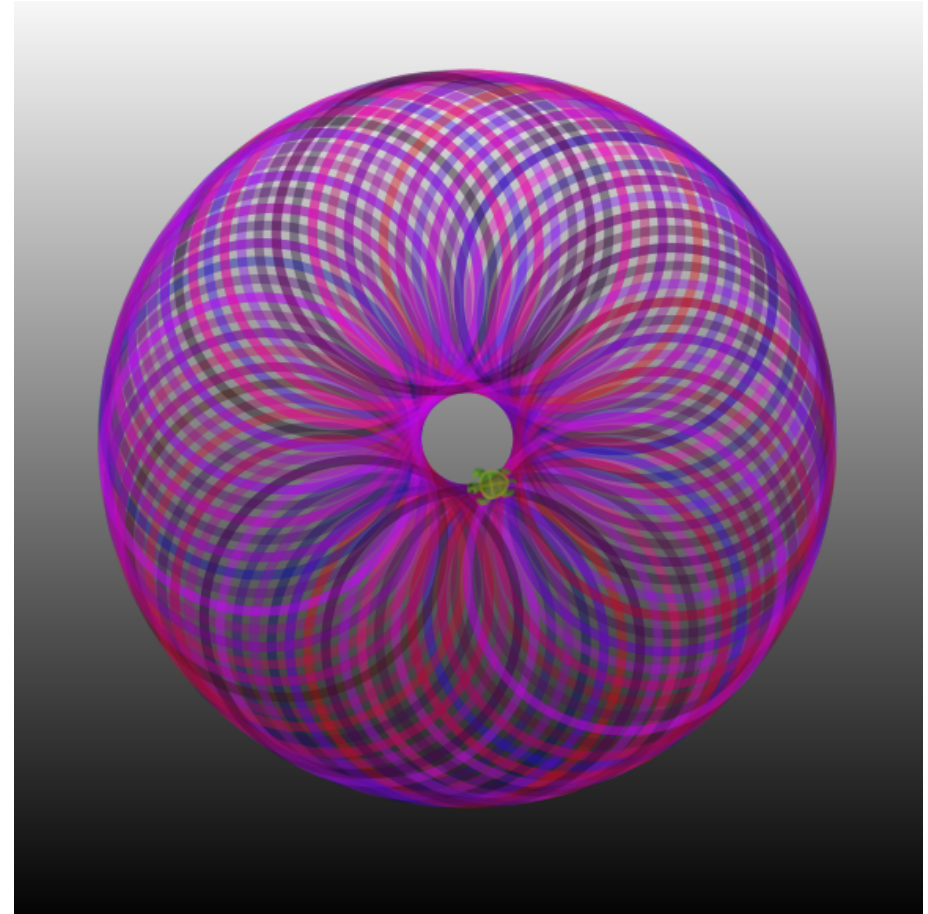
```
def random = random(256)
def randomColor = Color(random,10,random,100)

clear; setAnimationDelay(5)
setBackground2(black,white)
setPenThickness (6)

repeat(100) {
    setPenColor(randomColor)
    circle(100)
    hop(20)
    right(35)
}
```

## Challenge:

Try different random colors and backgrounds.

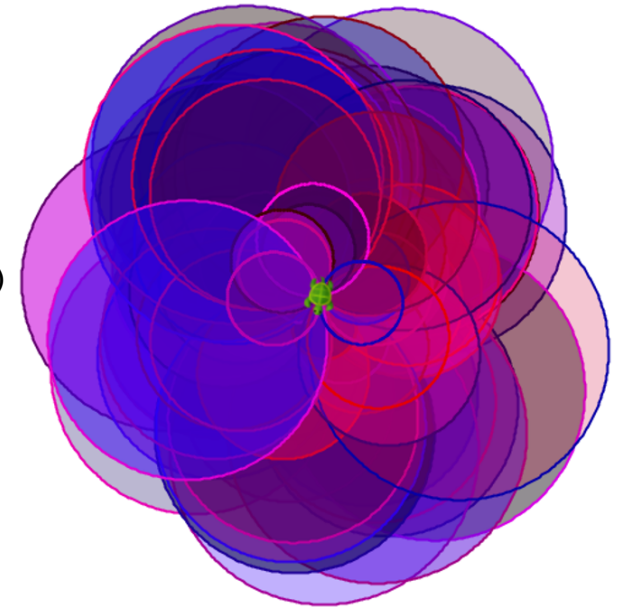


# Draw a flower

## Challenge:

The program below draws 100 random-colored circles, each at a random place with a random radius. Try to change the parameters and try to explain what happens.

```
clear(); setAnimationDelay(5)
setPenThickness (2)
repeat(100){
  setPenColor(Color(random(256),0,random(256)))
  setFillColor(Color(random(256),0,random(256),random(100)+50))
  left(random(360))
  circle(random(30)*4+10)
}
```



# Create a variable with **var**

With **var** you connect a name to a value.

You get a variable that you can assign a value like this:

```
var cucumber = 1  
cucumber = 1 + 1 //first calculate 1 + 1 and then assign that number to cucumber
```

## Challenge:

Try the program below. What does the turtle write?

```
var i = 0  
  
clear  
repeat(10){  
  i = i + 1  
  forward; write(i)  
}
```

## Tip:

- In the command `i = i + 1` `i` is given the new value which becomes the *old* value of `i` plus 1

# Draw many flowers

## Challenge:

- Make a function called `flower`, that paints a crown and a stalk from the crowns middle with a green leaf.
- Draw 5 flowers next to each other.

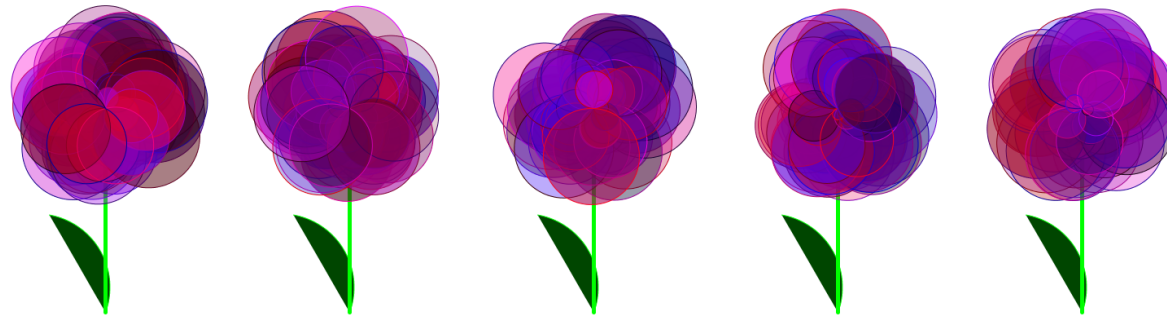
## Tip:

You can draw leafs with `arc(radius, angle)`.

Let the function `flower` have two parameters, `x` and `y`, and use `jumpTo(x,y)`

You can loop 5 times and calculate the position like this:

```
var i = 0
repeat(5){
  flower(600*i,0)
  i = i + 1
}
```



# Change the turtle's costume

## Challenge:

Download the mediafiles from Kojos homepage: [www.kogics.net/kojo-download#media](http://www.kogics.net/kojo-download#media)

- Unzip the file `scratch-media.zip` and find the crabpicture `crab1-b.png` in the folder `Media/Costumes/Animals`
- Put the file `crab1-b.png` in the same folder as your program.
- Try changing the costume of the turtle to a crab like this:

```
clear  
setCostume ("crab1-b.png")  
setAnimationDelay(2000)  
forward(1000)
```



## Tip:

- You can also use your own pictures of the types `.png` or `.jpg`
- If you want to put the picture in another folder you have to write the path to the file, for example `setCostume("~/Kojo/Media/Costumes/Animals/crab1-b.png")` where `~` means your homefolder.

# Make many turtles with **new**

You can create many new turtles with **new** like this:

```
clear
```

```
val p1 = new Turtle(100,100) //the new turtle p1 starts on position (100, 100)
```

```
val p2 = new Padda(100, 50) //the new turtle p2 starts on position (100, 50)
```

```
p1.forward(100)
```

```
p2.forward(-100) //turtle p2 backs up
```

## Challenge:

- Create three turtles that stand above each other.
- Make all their heads turn left.

## Tip:

- p1 and p2 are the turtles *names*. You can pick any name you want.
- With the name p1 and a dot you can give a specific turtles instructions, like this: p1.left
- invisible turns the ordinary turtle invisible.



# Make a turtle race

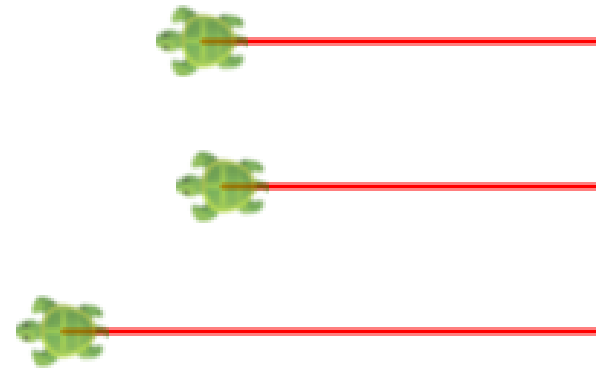
With the help of random number you can make the turtles race against each other.

## Challenge:

- Let three turtles race.
- Let all turtles run forward 10 times. Which turtle wins?

## Tip:

- With `p1.forward(random(100) + 1)` the turtle p1 moves 1 to 100 steps forward



# Alternative with **if**

With an **if**-command the computer chooses one of two different alternatives depending on a condition that can be true or false.

```
clear; invisible  
if (true) write("true") else write("false")
```

## Challenge:

- Change **true** to **false** and check what the turtle writes.
- Change the condition to  $2 > 1$  and check what the turtle writes.
- Change the condition to  $2 < 1$  and check what the turtle writes.
- Explain how an **if**-statement works.

## Tip:

- If the condition after **if** is **true** whatever is after that condition is picked.
- If the condition after **if** is **false** whatever is after **else** is picked.



# React to what the user is doing

```
clearOutput; setOutputTextFontSize(35)
val password = "cucumber"
val question  = "What is the password?"
val right     = "The safe is open!"
val wrong     = "You may not come in!"
val answer = readln(answer) //wait for an answer from the user
val message = if (answer == password) right else wrong
println(message)
```

## Challenge:

- Try the program and explain what happens.
- Change the password, question and what is printed when the question is right and wrong.
- Also ask for a user name and add it to what is printed.

# Make a **while**-loop

With a **while**-loop the computer will repeat a command as long as the condition is true.

```
clear; invisible; setAnimationDelay(250); clearOutput
var x = 200
while (x > 0) { //check the condition before each round
    forward(x); right
    write(x)
    x = x - 12
}
println("x is now: " + x)
```

## Challenge:

- What is printed in the output window? Why?
- Trace the program with the orange-colored play-button and check every step.
- Change the reduction of x from 12 to 20. Explain what happens.

# Guess the number

```
val secretNumber = random(100)+1
var answer = readln("Guess a number between 1 and 100! ")
var continue = true

while (continue) {
    if (answer.toInt < secretNumber)
        answer = readln(answer + " is too SMALL, guess again!")
    else if (answer.toInt > secretNumber)
        answer = readln(answer + " is too LARGE, guess again!")
    else if (answer.toInt == secretNumber)
        continue = false
}
println(secretNumber + " is the CORRECT answer!")
```

## Challenge:

Introduce a variable `var numberOfTries = 0` and count each try.  
When ready print the number of tries like this:  
Correct answer! You got it in 5 guesses

# Practice multiplication

```
var rightAnswers = 0
val startTime = System.currentTimeMillis / 1000
repeat(12) {
    val number1 = random(12)+1
    val number2 = random(12)+1
    val answer = readln("What is " + number1 + "*" + number2 + "?")
    if (answer == (number1 * number2).toString) {
        println("Correct!")
        rightAnswers = rightAnswers + 1
    }
    else println("Wrong. The right answer is " + (number1 * number2))
}
val stopTime = System.currentTimeMillis / 1000
val sec = stopTime - startTime
println("You got " + rightAnswers + " right answer in " + sec + " seconds")
```

## Challenge:

Change so that you only practice on multiplications of 8 and 9.

# Save the animals in a vector

```
var animal = Vector("elk", "cow", "rabbit", "mite") // the variable animal refers to a vector with 4 animals
println("The first animal in the vector is: " + animal(0)) //the positions in a vector are counted from 0
println("The second animal in the vector is: " + animal(1))
println("There are these many animals in the vector: " + animal.size)
println("The last animal in the vector is: " + animal(animal.size-1))
```

```
val s = random(animal.size) //take a random number between 0 and the number of animals minus 1
println("A random animal: " + animal(s))
animal = animal :+ "camel" //adds another animal last in the vector
animal = "dromedary" +: animal // adds another animal first in the vector
```

```
animal = animal.updated(2, "mudskipper") // Change the third animal(index 2 in vector)
println("All animals in the array backwards:")
animal.foreach{ x => println(x.reverse) } // for all x in array: type out x backwards.
```

## Challenge:

- What does the program print out in the output window? Explain what's happening.
- Add more animals to the array.

# Practice words

```
val Swedish = Vector("dator", "sköldpadda", "cirkel")
val English = Vector("computer", "turtle", "circle")
var amountRight = 0
repeat(5) {
    val s = random(3)
    val word = Swedish(s)
    val answer = readln("What is " + word + " in English?")
    if (answer == English(s)) {
        println("Correct answer!")
        amountRight = amountRight + 1
    } else {
        println("Wrong answer. Correct answer is: " + English(s))
    }
}
println("You have" + amountRight + " correct answers.")
```

## Challenge:

- Add more words.
- Practice words from English to Swedish.
- Let the user choose how many questions before end. Tip: `val amount = input("Amount: ").toInt`

# Capital Game

```
def capitalGame = {
  println("Welcome to the Capital Game!")
  val city = Map("Sweden" -> "Stockholm", "Denmark" -> "Copenhagen", "Skåne" -> "Malmö")
  var countriesLeft = city.keySet //keySet gives an amount of all keys in a Map
  def randomCountry = scala.util.Random.shuffle(countriesLeft.toVector).head
  while(!countriesLeft.isEmpty) {
    val country = randomCountry
    val answer = input("What is the capital in " + country + "?")
    output(s"You wrote: $answer")
    if (answer == city(country)) {
      output("Correct answer! You have " + countriesLeft.size + " countries left!")
      countriesLeft = countriesLeft - country //remove country from the set of countries left
    } else output(s"Wrong answer. The capital in $country begins with ${city(country).take(2)}...")
  }
  output("THANK YOU FOR PLAYING! (Press ESC)")
}

toggleFullScreenOutput;
setOutputBackground(black); setOutputTextColor(green); setOutputTextFontSize(30)
repeat(100)(output("")) //scroll the output window with 100 blank rows.
capitalGame

// *** TASK: (1) Add more pairs of countries and cities: country -> city (2) Measure time and count points.
```

# Make a timer with **object**

```
object timer {  
  def now = System.currentTimeMillis //gives time now in milliseconds.  
  var time = now  
  def reset = { time = now }  
  def measure = now - time  
  def randomWait(min: Int, max: Int) = //wait between min and max seconds  
    Thread.sleep((random(max-min)+min)*1000) //Thread.sleep(1000) waits 1 second  
}  
  
println("Click in the println window and wait...")  
timer.randomWait(3,6) //wait between 3 and 6 seconds  
timer.reset  
readln("Press Enter as fast as you can.")  
println("Reaction time: " + (timer.measure/1000.0) + " seconds")
```

With **object** you can collect things that belong together into an object.  
You can reach a thing inside an object with a dot: `timer.reset`

## Challenge:

- Try the program and measure your reaction time. How fast are you?
- Use timer in the task *Guess the number* and add the print-out:  
Correct answer! You made it in 5 guesses and 32 seconds



# Simulate a traffic light

```
def turnOffAll = draw(penColor(gray) * fillColor(black) -> PicShape.rect(130,40))
def light(c: Color, h: Int) = penColor(noColor) * fillColor(c) * trans(20,h) -> PicShape.circle(15)
def lightRed = draw(light(red, 100))
def lightYellow = draw(light(yellow, 65))
def lightGreen = draw(light(green, 30))
def wait(seconds: Int) = Thread.sleep(seconds*1000)

clear; invisible
while (true) { //an infinite loop
  turnOffAll
  lightRed; wait(3)
  lightYellow; wait(1)
  turnOffAll
  lightGreen; wait(3)
  lightYellow; wait(1)
}
```



## Challenge:

- How does the traffic light switch? Try to explain what happens.
- Change so that the green light is on for the double amount of time.

# Control the turtle with the keyboard

```
clear; setAnimationDelay(0)
activateCanvas()

animate { forward(1) }

onKeyPress { k =>
  k match {
    case Kc.VK_LEFT => left(5)
    case Kc.VK_RIGHT => right(5)
    case Kc.VK_SPACE => forward(5)
    case _ =>
      println("Another key: " + k)
  }
}
```

## Challenge:

- Write Kc. and press Ctrl+Alt+Space and look up what the different keys are called.
- Do penUp if you press the arrow up key
- Do penUp if you press the arrow down key
- Do color(blue) if you press B
- Do color(red) if you press R
- Increase or decrease the speed if you press + or -

# Control the turtle with the mouse

```
clear; setAnimationDelay(100)
activateCanvas()
```

```
var draw = true
```

```
onKeyPress { k =>
  k match {
    case Kc.VK_DOWN =>
      penDown()
      draw = true
    case Kc.VK_UP =>
      penUp()
      draw = false
    case _ =>
      println("Another key: " + k)
  }
}

onMouseClicked { (x, y) =>
  if (draw) moveTo(x, y) else jumpTo(x, y)
}
```

## Challenge:

- Do `setFillColor(black)` if you press the key F
- Introduce a variable `var fillNext = true` and in the case you press `Kc.VK_F` do:

```
if (fillNext) {
  setFillColor(black)
  fillNext=false
} else {
  setFillColor(noColor)
  fillNext=true
}
```

# Make your own bank account

```
object myAccount {  
  val number = 123456  
  var balance = 0.0  
  def in(amount: Double) = {  
    balance = balance + amount  
  }  
  def out(amount: Double) = {  
    balance = balance - amount  
  }  
  def showBalance() = {  
    println("Account number: " + number)  
    println("      Balance: " + balance)  
  }  
}
```

```
myAccount.showBalance()  
myAccount.in(100)  
myAccount.showBalance()  
myAccount.out(10)  
myAccount.showBalance()
```

## Challenge:

- What is the balance after the program has finished? Explain what's happening.
- Make it impossible to withdraw more than there is in the account.
- Add **val** maxAmount = 5000 and make sure you can't withdraw more than maxBelopp at a time.

# Make many objects from a **class**

A class is needed to make many accounts. With **new** new objects are made. Each object gets a number and balance.

```
class Account(number: Int) {  
  private var balance = 0.0 //private means "secret"  
  def in(amount: Double) = {  
    balance = balance + amount  
  }  
  def out(amount: Double) = {  
    balance = balance - amount  
  }  
  def showBalance() =  
    output(s"Account $number: $balance")  
}  
  
val account1 = new Account(12345) //new makes an object  
val account2 = new Account(67890) //another object
```

```
account1.in(99)  
account2.in(88)  
account1.out(57)  
account1.showBalance  
account2.showBalance
```

## Challenge:

- What is the balance on the different accounts when the program has finished? Explain what happens.
- Make even more Account objects and deposit and withdraw money from these.
- Add a class parameter name: String containing the name of the owner of the bank account.
- Do that even the name is printed when showBalance is called
- What happens if you do:  
account1.balance = 10000000

# Talk to the computer

```
setOutputBackground(black); setOutputTextFontSize(30); setOutputTextColor(green)
println("Write interesting answers even if the questions are weird. End with 'good bye'")
def randomize(xs: Vector[String]) = scala.util.Random.shuffle(xs).head
val text = Vector("What does this mean: ", "Do you like", "Why is this needed: ", "Tell more about")
var answer = "?"
val opening = "What do you want to talk about?"
var word = Vector("bellybutton fluff", "ketchup-icecream", "Santa Claus", "pillow")
while (answer != "good bye") {
  val t = if (answer == "?") opening
    else if (answer == "No") "Well, no."
    else if (answer == "Yes") "Well, yes."
    else if (answer.length < 4) "Okay..."
    else randomize(text) + " " + randomize(word) + "?"
  answer = readln(t).toLowerCase
  word = word ++ answer.split(" ").toList.filter(_.length > 3)
}
println("Thanks for the talk! Now I know these words:" + word)
```

//Task:

// (1) Try the program and explain what's happening.

// (2) When does the while-loop finish?

// (3) Add more strings in the vectors "text" and "word".

// (4) Add more good answers to short words apart from "No" and "Yes".

# Modify the pong game

## Challenge:

- Choose the menu Samples > Animations and Games > Pong and try the game.
- You control with the up and down arrows, A and Z.
- Press ESC to cancel the game and examine the code.
- Change the code so that the ball becomes bigger.
- Turn the playing field into a tennis field, with green background, white lines and a yellow ball.

