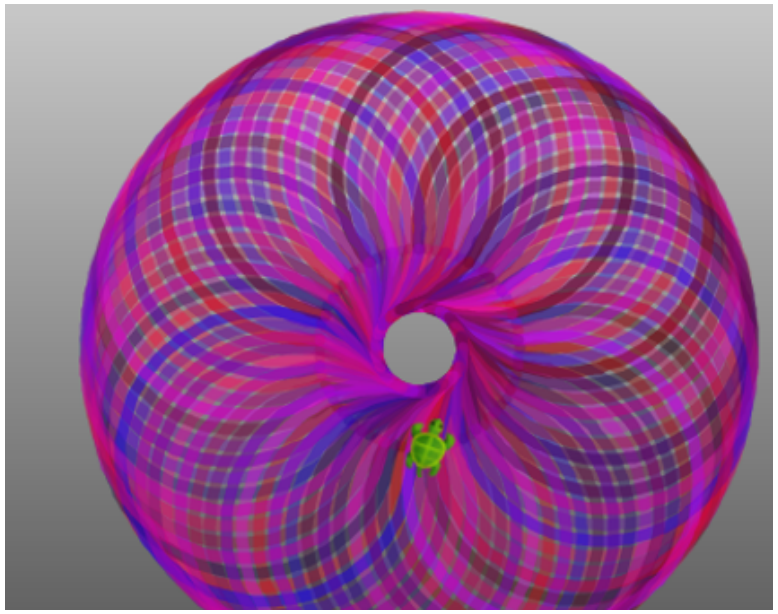
Sfide con Kojo

Editor: Björn Regnell
www.lth.se/code



Challenges with Kojo

Version: 15 aprile 2016

Editor: Björn Regnell
Traduzione in italiano: Massimo Maria Ghisalberti
Contributors: Björn Regnell, Lalit Pant, Sandra Nilsson, Maja Johansson, Simone Strippgen, Christoph Knabe, Massimo Maria Ghisalberti

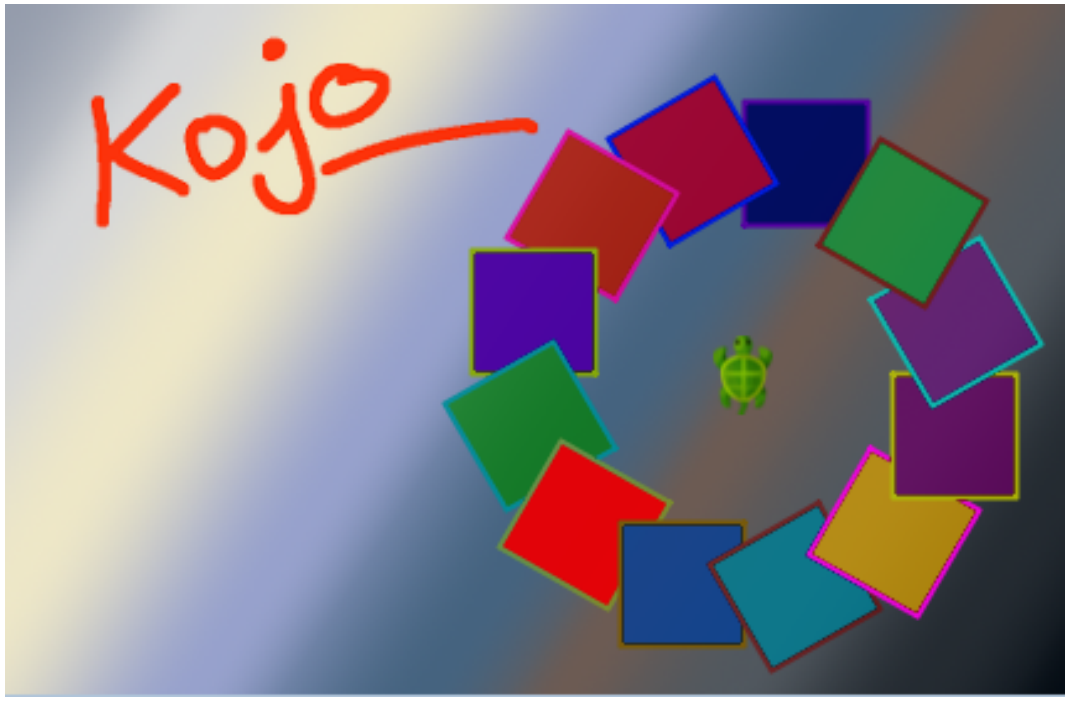
`http://lth.se/programmera`

Indice

Su Kojo

Che cosa è Kojo?

Kojo è una applicazione che può aiutare ad imparare a programmare un computer. Con Kojo si può scrivere del codice in un moderno ed estremamente potente linguaggio di programmazione **Scala**. Kojo è Open Source ed è libero e disponibile per Linux, Windows e Mac.



Dove posso trovare Kojo?

Kojo può essere scaricato qui:
www.kogics.net/kojo-download
Per più informazioni:
www.kogics.net/kojo

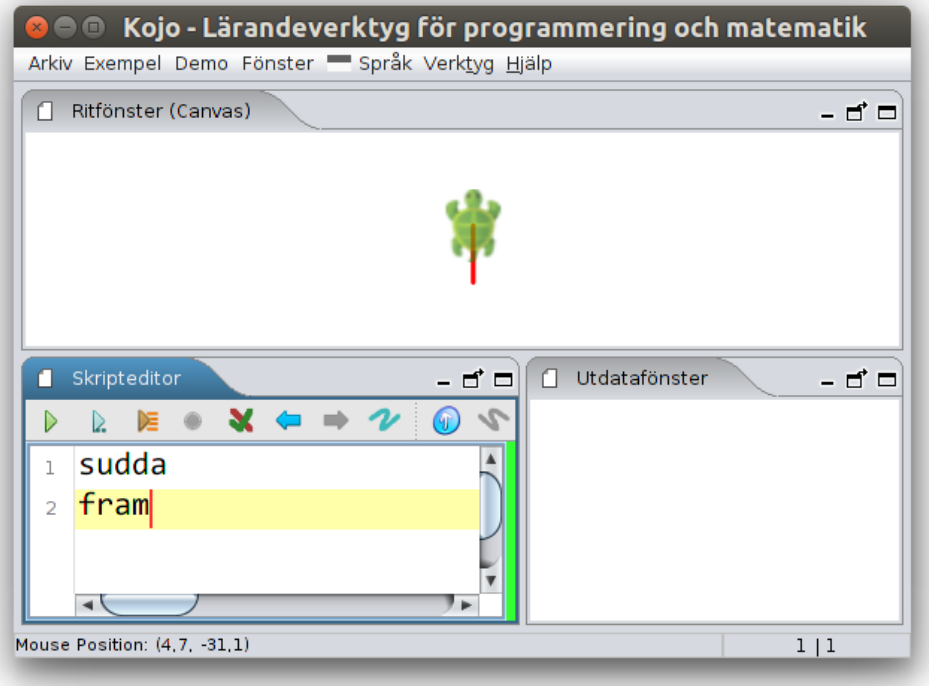
Il vostro primo programma

Sfida:

Scrivete quello che segue nell'area del codice:

```
pulisci
avanti
```

Premete il bottone verde  per eseguire il codice.



Disegnamo un quadrato

```
pulisci
avanti
destra
```

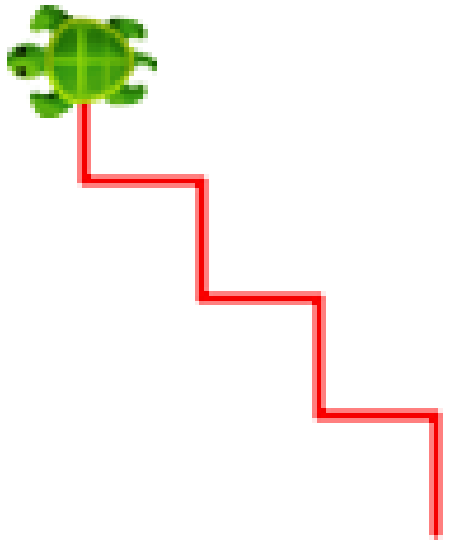
Scrivendo `sinistra` o `destra` la tartaruga cambierà direzione.

Sfida:

Estendete il programma in moda da fare disegnare un quadrato.

Disegnamo delle scale

```
pulisci
avanti; sinistra
avanti; destra
```



Con il punto e virgola ; tra le istruzioni, si possono avere più comandi sulla stessa linea.

Sfida:

Estendete il programma in maniera che disegni delle scale.

Facciamo un ciclo

```
pulisci
ripeti(4){ avanti; destra }
```

Sfida:

- Cosa capiterà se cambiamo 4 in 100?
- Disegnerà delle scale con 100 gradini.

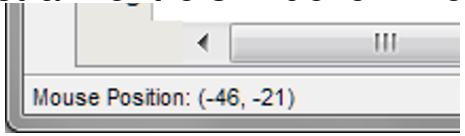
Disegnamo un personaggio

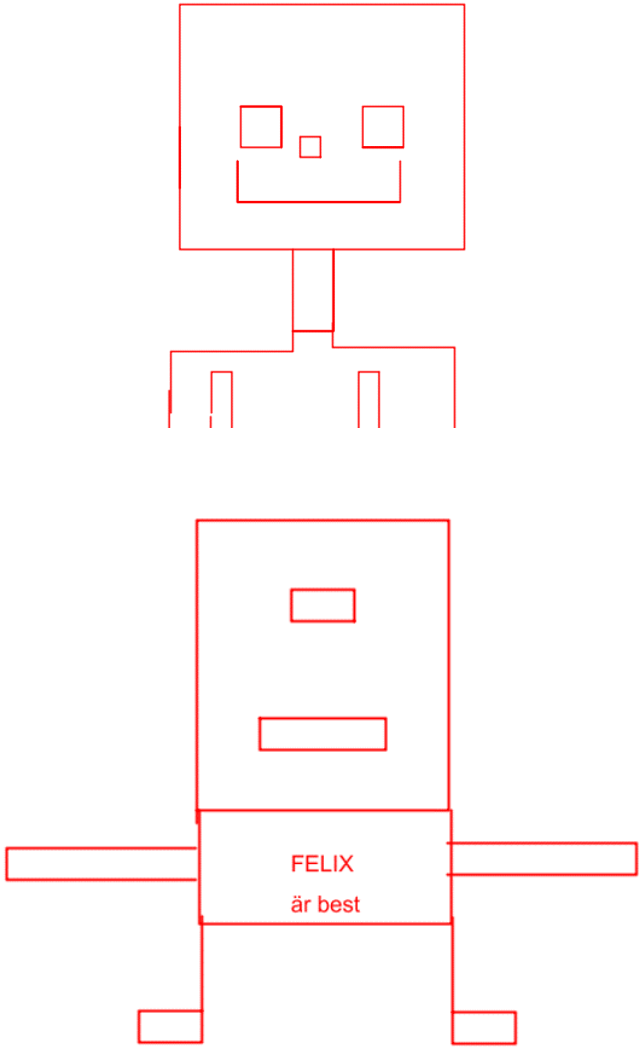
Sfida:

Disegnate un personaggio di vostra scelta.

Tip:

```
salta
sinistra(180)
avanti(300)
salta(100)
saltaVerso(25,-28)
scrivi("FELIX is awesome")
colorePenna(purple)
coloreRiempimento(green)
```

Si può vedere la posizione della tartaruga in basso sulla sinistra mentre si muove il mouse sull'area di disegno: 



Quanto è veloce il vostro computer?

Il primo elaboratore elettronico era chiamato **ENIAC** e poteva contare fino a 5000 in un secondo.
In Kojo c'è una funzione `räknaTill` che misura quanto veloce il vostro computer conti.
Eseguendo `conta(5000)` sul mio computer più veloce, appare questa scritta nell'area di output:

```
*** 5000 *** PRONTO!
Ci sono voluti 0.32 millisecondi.
```

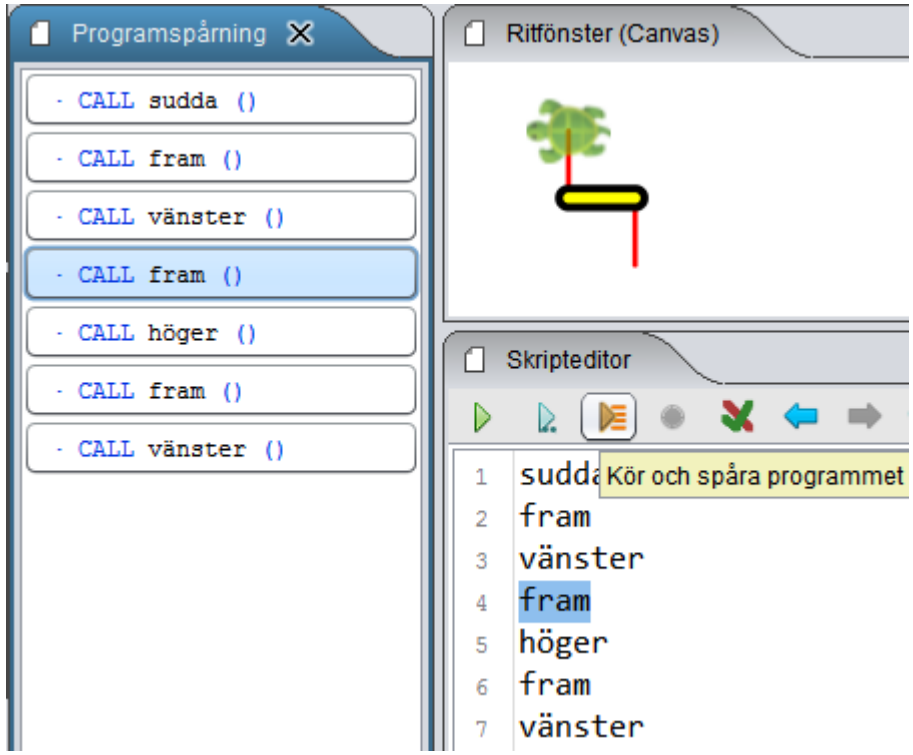
Sfida:

- Eseguite `conta(5000)` e controllate se il vostro computer è più veloce del mio.
- Quando tempo tempo ci metterà il vostro computer a contare fino ad un milione?
- Fino a che numero può contare in un secondo?

Tracciamo l’esecuzione del programma

Sfida:

- Scriviamo un programma che disegna scale.
- Premiamo il bottone arancione.
- Premiamo su uno dei comandi: `CALL fram`. Cosa succede nell’area di disegno?
- Quando una parte del codice è marcata in blu, solo quella parte verrà eseguita premendo il bottone di avvio. Possiamo deselezionare il codice facendo click dopo il codice che è selezionato.
- Aggiungete altri comandi al vostro programma ed osservate cosa accade quando lo tracciate.
- Chiudete la *Program trace* area quando avrete fatto.



Scriviamo le nostre funzioni con `def`

Con **`def`** si possono scrivere le proprie *funzioni* scegliendone il nome.

```
def quadrato = ripeti(4){ avanti; destra }

pulisci
quadrato   //use your square-function
salta
quadrato
```

Sfida:

- Cambiate il colore del quadrato.
- Fatelo varie volte.

Tip:

```
coloreRiempimento(green); colorePenna(purple)
```

Una pila di quadrati

Sfida:

Facciamo una pila di 10 quadrati

Tip:

```
def quadrato = ripeti(4){ avanti; destra }

pulisci; ritardo(100)
ripeti(10){ ??? }
```

Una funzione per fare le pile

Sfida:

Scrivete una funzione chiamata `pila`, che disegna una pila di 10 quadrati.

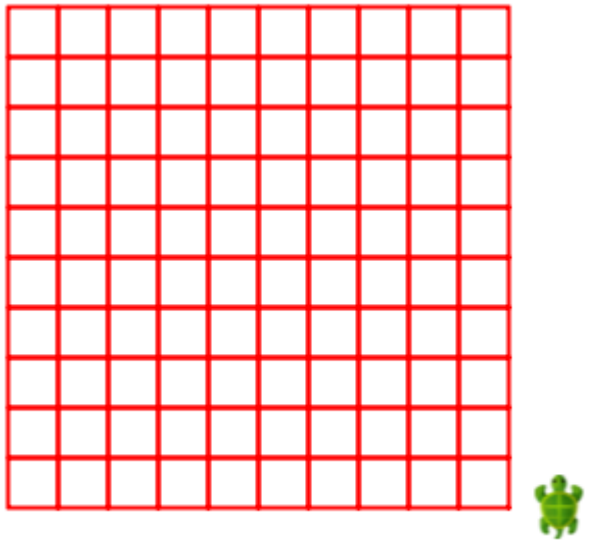
Tip:

```
def quadrato = ripeti(4){ avanti; destra }
def pila = ???

pulisci; ritardo(100)
pila
```

Facciamo una griglia

Sfida:

Fate una griglia 10*10 di quadrati.



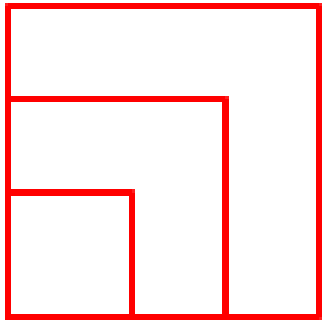
Tip:

- Usate la vostra funzione per le pile (stack) che avete scritto prima.
- Saltate indietro di una inter colonna con `salta(-10 * 25)`
- Saltate nella giusta posizione con `destra; salta; destra`

Un quadrato parametrico

Sfida:

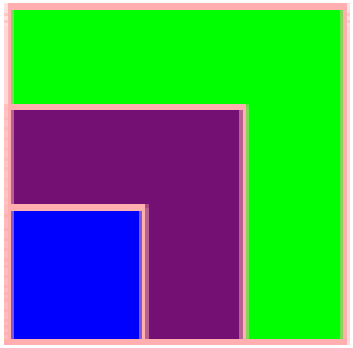
DIsegnamo un quadrato di dimensioni differenti.



Tip:

Date alla vostra funzione un *parameter*,
chiamato `side` di tipo `Int`:

```
def quadrato(side : Int) =
  ripeti(4){ avanti(side); destra }

pulisci; ritardo(100); invisibile
quadrato(100)
quadrato(70)
quadrato(40)
```

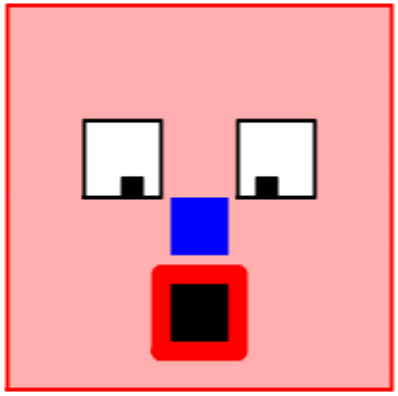


Potete cambiare il colore con:
`coloreRiempimento(blue); colorePenna(pink)`

Disegnamo un personaggio a quadrati

Sfida:

Disegnate un personaggio con quadrati di dimensione differente.



Tip:

```
def quadrato(x: Int, y: Int, side: Int) = {
  saltaVerso(x, y)
  ripeti(4) { avanti(side); destra }
}
def testa(x: Int, y: Int) = { coloreRiempimento(pink); colorePenna(red); quadrato(x, y, 200) }
def occhio(x: Int, y: Int) = { coloreRiempimento(white); colorePenna(black); quadrato(x, y, 40) }
def pupilla(x: Int, y: Int) = { coloreRiempimento(black); colorePenna(black); quadrato(x, y, 10) }
def naso(x: Int, y: Int) = { coloreRiempimento(blue); colorePenna(noColor); quadrato(x, y, 30) }
def bocca(x: Int, y: Int) = { impostaSpessorePenna(10); coloreRiempimento(black); colorePenna(red); quadrato(x

pulisci; ritardo(20); invisibile
testa(0, 0)
occhio(40, 100); pupilla(60, 100)
???
```

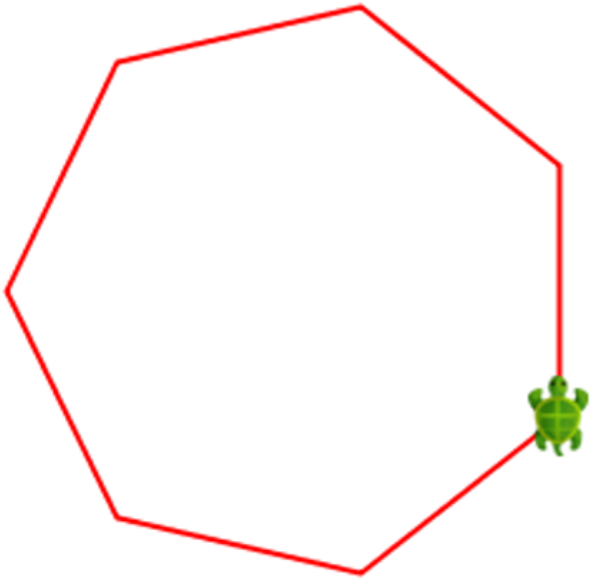
Disegnamo un poligono

Sfida:

- Provate il codice qui sotto. DIsegnate diversi tipi di poligoni.
- Aggiungete un parametro `side` e disegnate dei poligoni di dimensioni differenti.
- Quanto dovrebbe essere largo n per farlo sempreare un cerchio?

Tip:

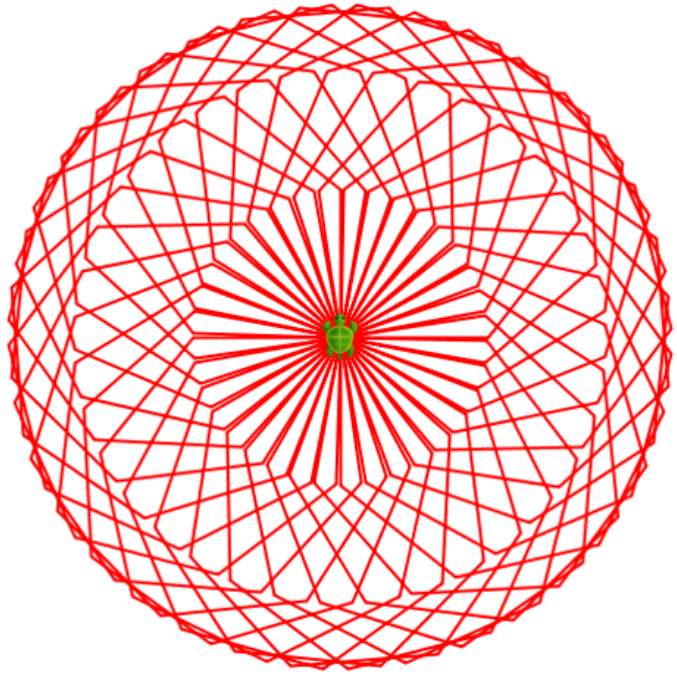
```
def poligono(n:Int) = ripeti(n){
  avanti(100)
  sinistra(360.0/n)
}

pulisci; ritardo(100)
poligono(7)
```



Disegnamo alcuni poligoni

Sfida:

- Provate il codice qui sotto.
- Cercate di cambiare il numero di lati e di angoli.
- Colorate i poligoni in colori differenti.



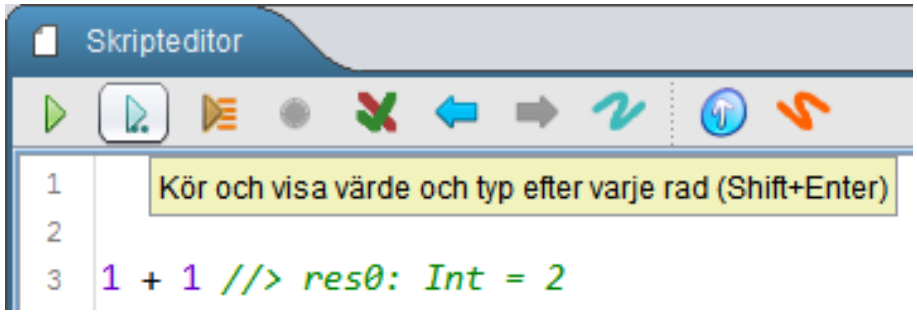
```
def poligono(n: Int, side: Int) = ripeti(n){
  avanti(side)
  sinistra(360.0/n)
}
def ruota(n: Int, heading: Int, side: Int) =
  ripeti(360/heading){ poligono(n, side); sinistra(heading) }

pulisci; ritardo(5)
ruota(7, 10, 100)
```

Valori ed espressioni

Sfida:

- Scrivete `1 + 1` e premete il bottone blu. Kojo creerà un commento verde.
- Il commento mostra il valore dell'espressione `1 + 1` che è equivalente a `2` ed il cui tipo è `Int`, che significa numero intero.
- Scrivete altre espressioni. Che valori e che tipi hanno le espressioni qui sotto?

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



Tip:

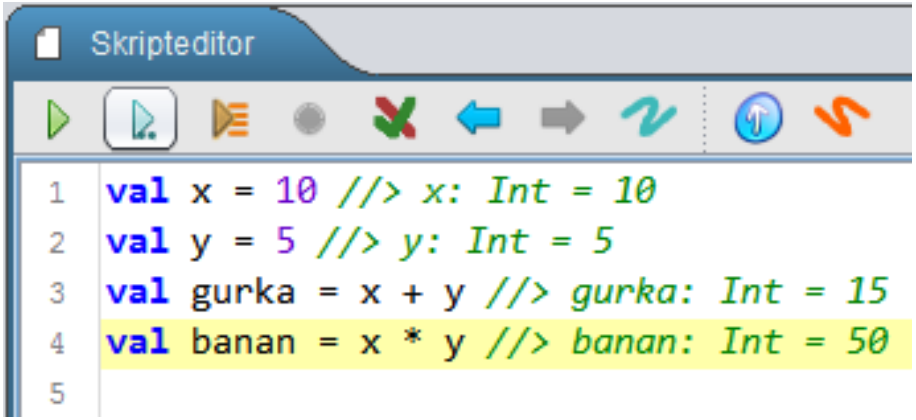
- `/` tra numeri interi la divisione ignora i valori decimali (divisione intera). Per essere sicuri che la divisione non sia di tipo intero uno dei due operandi deve avere numeri decimali. Il tipo di un numero decimale è chiamato `Double`.
- Con `%` si può avere il resto di una divisione intera.

Diamo un nome ad un valore con `val`

Sfida:

Con `val` si può riferire un nome ad un valore. Si può usare il nome al posto del valore. Provate il programma sotto. Cosa scriverà la tartaruga?

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

pulisci
avanti; scrivi(banana)
avanti; scrivi(cocomero)
avanti; scrivi(y)
avanti; scrivi(x)
```

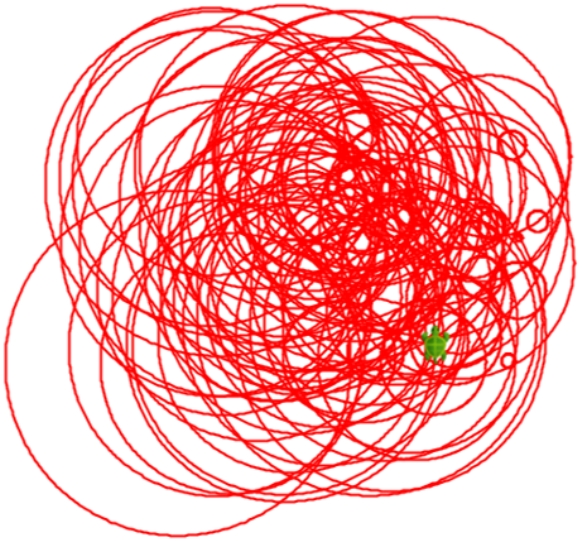


Numeri casuali

Sfida:

- Fate eseguire il programma sotto varie volte, cose succede?
- Quale è il più piccolo ed il più grande valore possibile del raggio `r`?
- Cambiate il raggio così che `r` diventi un numero casuale compreso tra 3 e 200.
- Disegnate 100 cerchi, ognuno con un raggio casuale ad una posizione casuale come mostrato nella figura.

```
//r becomes a random number between 10 and 99:
val r = numeroCasuale(90) + 10

pulisci; ritardo(10); invisibile
scrivi("Radius = " + r)
cerchio(r)
```



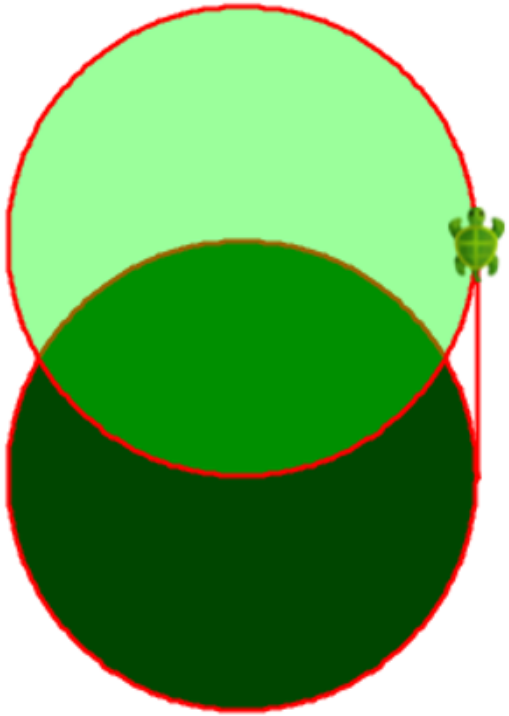
Misceliamo i nostri colori

- Potete mischiare i vostri colori con `Color`, per esempio `Color(0, 70, 0)`
- I tre parametri sono i valori per *red*, *green* e *blue*
- Potete aggiungere un quarto parametro che imposterà la *transparency*
- L'intervallo per i parametri è un numero intero compreso tra 0 e 255

Sfida:

Provate il programma sotto e cambiate la trasparenza del colore

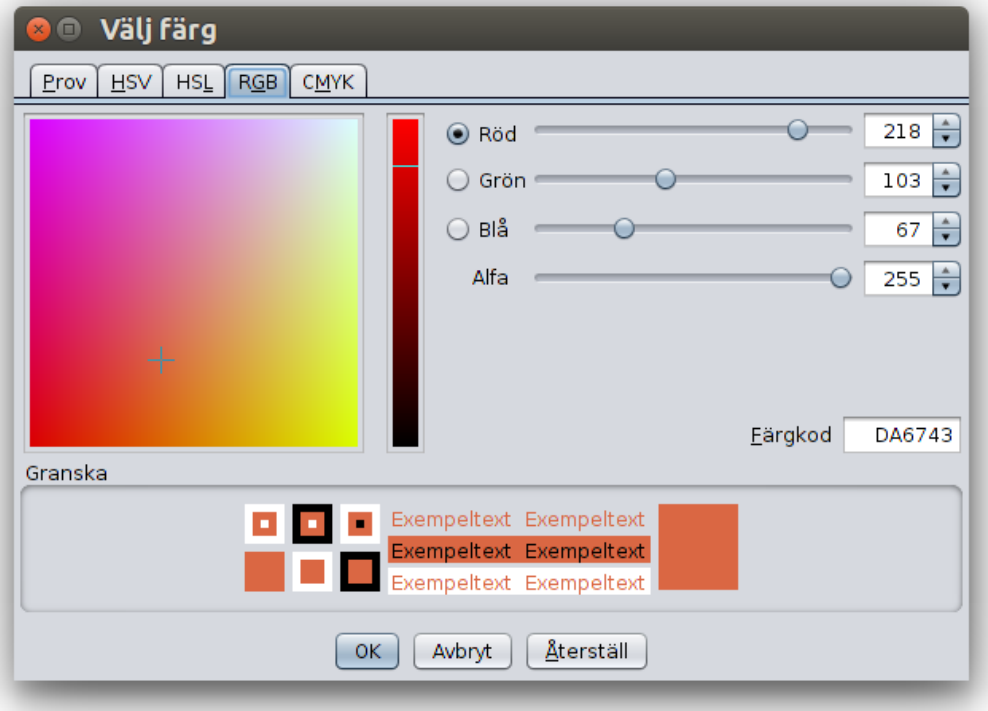
```
pulisci; ritardo(100)

val verdeOliva = Color(0,70,0)
val pistacchio = Color(0,255,0,100)

coloreRiempimento(verdeOliva); cerchio(100)
coloreRiempimento(pistacchio); avanti(100); cerchio(100)
```



Proviamo il selettore dei colori

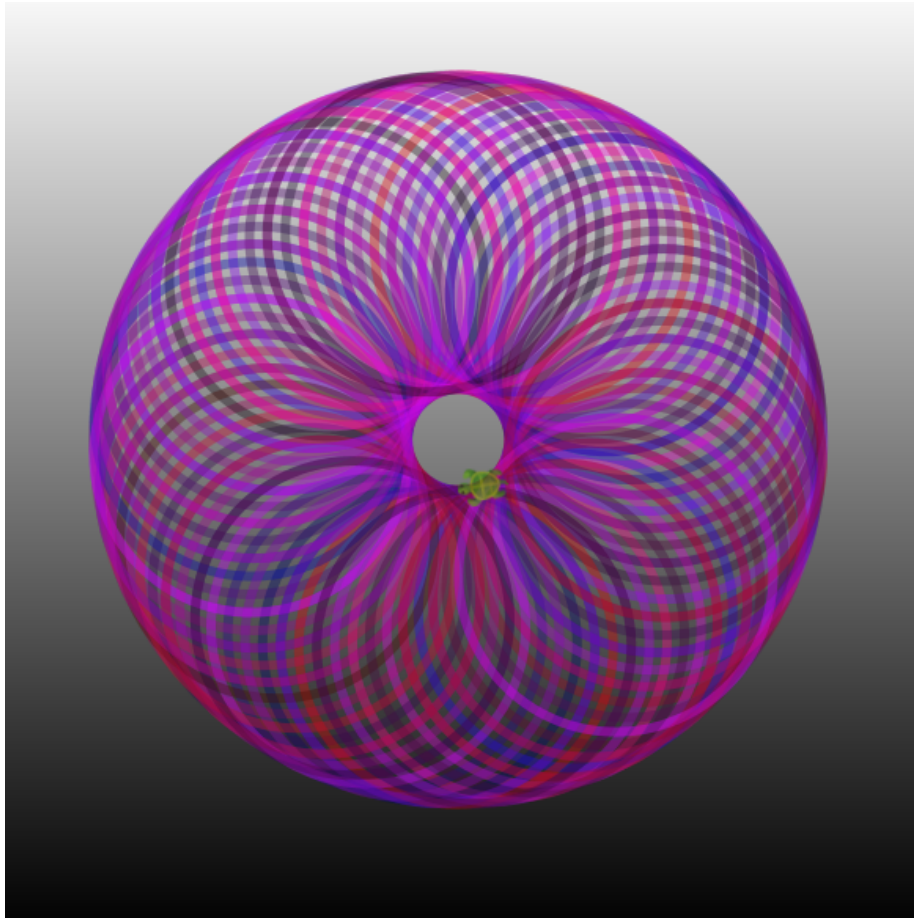
Sfida:

- Fate click con il tasto destro del mouse nell'area del codice e selezionate `Choose color...`
- Se scegliete la linguetta **RGB** potrete scegliere un nuovo colore in valori RGB.
- Premete Ok e guardate nell'area di output. Potrete notare i valori RGB pe il rosso, il verde ed il blu.
- Potete usare questi valori nei vostri programmi per disegnare con il vostro nuovo colore, per esempio in questo modo: `colore(Color(218,153,67))`.



Disegnamo dei cerchi casuali

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

pulisci; numeroCasuale(5)
gradiente(black,white)
impostaSpessorePenna (6)

ripeti(100) {
    colorePenna(randomColor)
    cerchio(100)
    salta(20)
    destra(35)
}
```



Sfida:

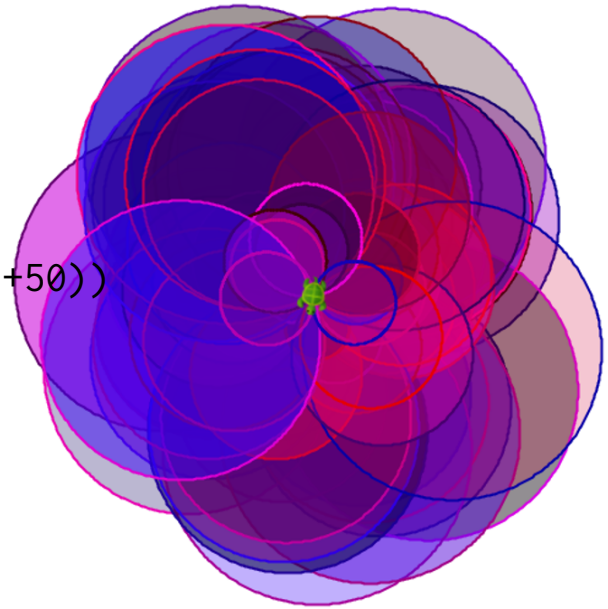
Provate differenti colori e sfondi casuali.

Disegnamo un fiore

Sfida:

Il programma qui sotto disegna 100 cerchi colorati casualmente, ognuno con posizione e raggio casuale. Cambiate i parametri e cercate di spiegare che succede.

```
pulisci(); ritardo(5)
impostaSpessorePenna (2)
ripeti(100){
  colorePenna(Color(random(256),0,random(256)))
  coloreRiempimento(Color(random(256),0,random(256),random(100)+50))
  sinistra(random(360))
  cerchio(random(30)*4+10)
}
```



Creiamo una variabile con `var`

Con **`var`** si può associare un nome ad un valore, ma questo può essere cambiato in seguito.
Potete prendere la variabile ed assegnargli un valore in questo modo:

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

Sfida:

Provate questo programma. Cosa scriverà la tartaruga?

```
var i = 0

pulisci
ripeti(10){
  i = i + 1
  avanti; scrivi(i)
}
```

Tip:

- Nella espressione `i = i + 1` ad `i` è stato assegnato il *old* valore di `i` più 1

Disegnamo alcuni fiori

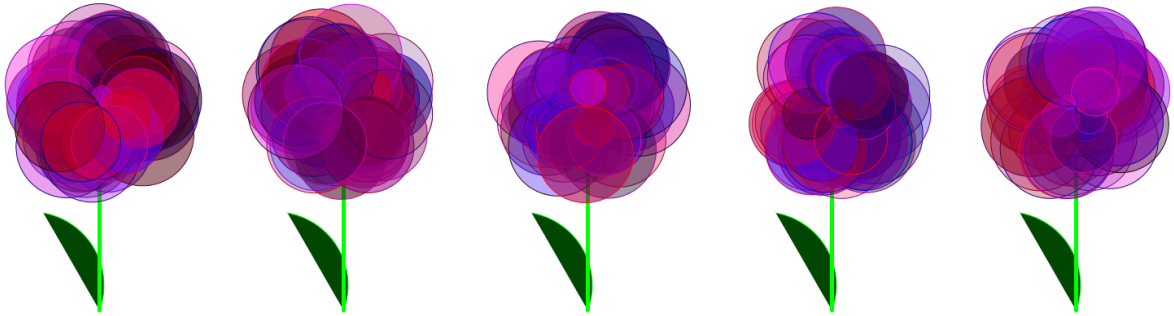
Sfida:

- Fate una funzione chiamata `flower`, che disegna una corolla da cui parte un ramo con una foglia verde.
- Disegnate 5 fiori uno accanto all'altro.

Tip:

Potete disegnare le foglie con `arco(radius, angle)`.
Fate che la funzione `fiore.` abbia due parametri, x e y, e usate `saltaVerso(x,y)`
Potete ripetere il ciclo 5 volte e calcolare la posizione in questo modo:

```
var i = 0
ripeti(5){
  fiore(600*i,0)
  i = i + 1
}
```



Cambiamo l'immagine della tartaruga

Sfida:

Scaricate i file aggiuntivi dalla pagin web di Kojos: www.kogics.net/kojo-download#media

- Decomprimete il file `scratch-media.zip` e cercate l'immagine del granchio `crab1-b.png` nella cartella `Media/Costumes/Animals`
- Posizionate il file `crab1-b.png` nella stessa cartella del programma.
- Cercate di cambiare l'immagine della tartaruga in un granchio in questo modo:

```
pulisci
indossaCostume ("crab1-b.png")
ritardo(2000)
avanti(1000)
```

Tip:

- Potete usare anche delle vostre immagini, basta che siano del tipo `.png` o `.jpg`

- Se volte mettere le immagini in una cartella diversa da quella del programma, dovrete fornire il percorso sul disco rigido dove poter rintracciare il file, per esempio `indossaCostume("~/Kojo/Media/Costumes/Animals/crab1-b.png")` dove ~ significa la vostra cartella personale (la cartella home per i sistemi operativi di tipo Unix).

Facciamo una nuova tartaruga con new

Potere creare altre tartarughe con il comando **new** in questo modo:

```
pulisci
val p1 = new Tartaruga(100,100) //the new turtle p1 starts on position (100, 100)
val p2 = new Tartaruga(100, 50) //the new turtle p2 starts on position (100, 50)
p1.avanti(100)
p2.avanti(-100) //turtle p2 backs up
```

Sfida:

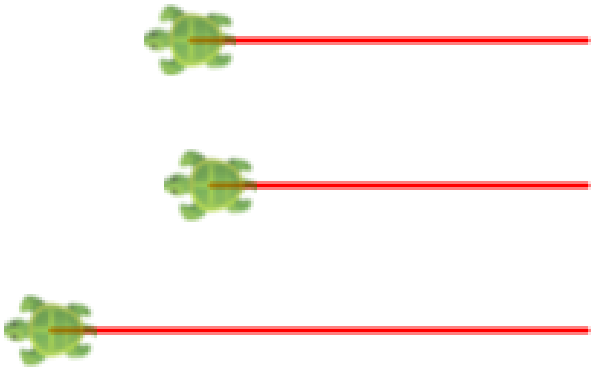
- Create tre tartarughe che siano posizionate una vicino all'altra.
- Fatele girare verso sinistra.

Tip:

- `p1` e `p2` sono i *names* delle tartarughe. Potete farne quante volete.
- Con il nome `p1` ed un punto, potete dare alle specifiche tartarughe dei comandi, per esempio così: `p1.sinistra`
- `invisibile` nasconde le tartarughe.

Una corsa di tartarughe

Con l'aiuto dei numeri casuale potrete programmare una corsa di tartarughe.



Sfida:

- Impostate una corsa tra tre tartarughe.
- Fate che corrano in avanti per 10 volte. Quale vincerà?

Tip:

- Con `p1.avanti(random(100) + 1)` la tartaruga p1 muoverà da 1 a 100 passi in avanti

Le scelte alternative con `if`

Con una istruzione **`if`** il computer sceglierà una delle due differenti alternative in dipendenza da una condizione che può risultare vera o falsa.

```
pulisci; invisibile
if (true) scrivi("true") else scrivi("false")
```

Sfida:

- Cambiate **`true`** in **`false`** e controllate cosa scrive la tartaruga.
- Cambiate la condizione in `2 > 1` e controllate cosa scrive la tartaruga.
- Cambiate la condizione in `2 < 1` e controllate cosa scrive la tartaruga.
- Spiegate come l'espressione **`if`** funziona.

Tip:

- Se la condizione dopo **`if`** è **`true`** viene preso quello subito dopo.
- Se la condizione dopo **`if`** è **`false`** viene preso quello dopo **`else`**.

Reagire a quello che l'utente sta facendo

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

Sfida:

- Eseguire il programma e spiegare cosa stia facendo.
- Cambiare la password, la domanda e cosa è stampato quando la risposta è giusta o sbagliata.
- Chiedere anche il nome dell'utente ed aggiungerlo a quello che viene scritto.

Facciamo un ciclo `while`

Con il ciclo while `while` il computer ripeterà un comando tante volte finché la condizione sarà vera.

```
pulisci; invisibile; ritardo(250); pulisciOutput
var x = 200
while (x > 0) { //check the condition before each round
  avanti(x); destra
  scrivi(x)
  x = x - 12
}
scriviLinea("x is now: " + x)
```

Sfida:

- Cosa viene scritto nell'area di output? Perché?
- Tracciate il programma con il bottone arancione di esecuzione e controllate ogni passo.
- Cambiate la riduzione di `x` da `12` a `20`. Spiegate cosa succede.

Indovina il numero

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

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

Sfida:

Introducete una variabile `var numberOfTries = 0` e contate ad ogni tentativo.
Quando pronti scrivete il numero dei tentativi in questo modo:
`Correct answer! You got it in 5 guesses`

Fare pratica nella moltiplicazione

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

Sfida:

Cambiate per poter fare pratica solo nella moltiplicazione con 8 e 9.

Immagazzinare gli animali in una lista

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

val s = numeroCasuale(animal.size) //take a random number between 0 and the number of animals minus 1
scriviLinea("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)
scriviLinea("All animals in the array backwards:")
animal.foreach{ x => scriviLinea(x.reverse) } // for all x in array: type out x backwards.
```

Sfida:

- Cosa stampera il programma nell'area di output? Spiegate cosa succede.
- Aggiungete altri animali alla lista.

Fare pratica nelle parole

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

Sfida:

- Aggiungete altre parole.
- Fare pratica nelle parole dall'inglese all'italiano.
- Fate scegliere più domande prima di finire. Suggerimento: `val amount = input("Amount: ").toInt`

Il gioco delle Capitali

```
def capitalGame = {
  scriviLinea("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)
ripeti(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.
```

Fare un timer con `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((numeroCasuale(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")
```

Con **`object`** è possibile raccogliere cose che sono in relazione tra loro in un oggetto.
Potete raggiungere una cosa all'interno di un oggetto con un punto: `timer.reset`

Sfida:

- Provate il programma e misurate il tempo di reazione. Quanto siete veloci?
- Usate `timer` nel compito *Guess the number* ed aggiungete un modo per scrivere:
  `Correct answer! You made it in 5 guesses and 32 seconds`

Simulazione di un semaforo

```
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)

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

Sfida:

- Cosa succede quando il semaforo cambia colore? Cercate di spiegare cosa succede.
- Fate una modifica in modo che la luce verde sia accesa per il doppio del tempo.

Controllare la tartaruga con la tastiera

```
pulisci; ritardo(0)
activateCanvas()

animate { avanti(1) }

onKeyPress { k =>
  k match {
    case Kc.VK_LEFT => sinistra(5)
    case Kc.VK_RIGHT => destra(5)
    case Kc.VK_SPACE => avanti(5)
    case _ =>
      scriviLinea("Another key: " + k)
  }
}
```

Sfida:

- Scrivete `Kc.` e premete `Ctrl+Alt+Space` e guardate i diversi tasti premuti.
- Chiamare `alzaPenna` premento freccia su
- Chiamare `abbassaPenna` premento freccia giù
- FaChiamarete `color(blue)` premendo il tasto B
- Chiamare `color(red)` premendo il tasto R
- Aumentare o diminuire la velocità premendo `+` o `-`

Controllare la tartaruga col mouse

```
pulisci; ritardo(100)
activateCanvas()

var draw = true

onKeyPress { k =>
  k match {
    case Kc.VK_DOWN =>
      abbassaPenna()
      draw = true
    case Kc.VK_UP =>
      alzaPenna()
      draw = false
    case _ =>
      scriviLinea("Another key: " + k)
  }
}

onMouseClick { (x, y) =>
  if (draw) moveTo(x, y) else saltaVerso(x, y)
}
```

Sfida:

- Chiamare `coloreRiempimento(black)` premenso il tasto F
- Introdurre la variabile `var fillNext = true` e nel caso sia premuto `Kc.VK_F` eseguire:

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

Fatevi il vostro conto in banca

```
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() = {
    scriviLinea("Account number: " + number)
    scriviLinea("     Balance: " + balance)
  }
}

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

Sfida:

- Qual'è il bilancio dopo che il programma è terminato? Spiegate cosa è successo.
- Rendete impossibile ritirare più del contenuto del conto.
- Aggiungete `val maxAmount = 5000` e fate in modo che non si possa ritirare più di `maxBelopp` alla volta.

Create molti oggetti da una `class`

C'è bisogno di dichiarare una classe per poter costrure molti conti. Con **new** sono creati nuovi oggetti di quel tipo. Ogni oggetto avra un suo numero ed un suo bilancio.

```
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
```

Sfida:

- Qual'è il bilancio dei differenti conti quando il programma avrà terminato l'esecuzione? Che cosa è successo.
- Fate altri conti depositando e prelevando denaro da questi.
- Aggiungete un parametro alla classe `name: String` che conterrà il nome del possessore del conto in banca.
- Fate che il `name` venga scritto quando `showBalance` è invocato
- Che succede se impostate: `account1.balance = 10000000`

Parliamo col computer

```
setOutputBackground(black); setOutputTextFontSize(30); setOutputTextColor(green)
scriviLinea("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 = leggiLinea(t).toLowerCase
  word = word ++ answer.split(" ").toList.filter(_.length > 3)
}
scriviLinea("Thanks for the talk! Now I know these words:" + word)

//Task:
// (1) Far eseguire il programma e spiegare che èsuccesso.
// (2) Quando il ciclo while èfinito che fa?
// (3) Aggiungete altro testo nelle liste "text" e "word".
// (4) Aggiungete altre buone risposte a parte "no" e "si".
```

Modifichiamo il gioco del ping pong

Sfida:

- Scegliere dal menu Esempi > Animazioni e giochi > ping pong. Provate a giocare
- Si controlla con i testi freccia su e freccia giù per il giocatore destro e A e Z per il sinistro.
- Si preme ESC per fermare il gioco ed esaminare il codice.
- Cambiare il codice per fare la palla più grande.
- cambiare il campo in un campo da tennis, con il prato verde, linee bianche ed una palla gialla.