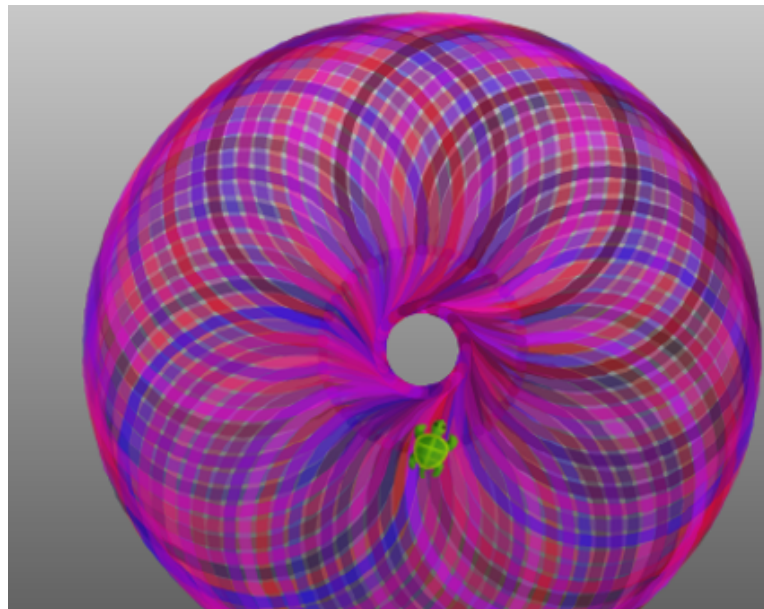


Uppdrag med Kojo

Redaktör: Björn Regnell
www.lth.se/programmera



Uppdrag med Kojo

Version: 28 februari 2015



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

Editor: Björn Regnell

Contributors: Björn Regnell, Lalit Pant, Sandra Nilsson, Maja Johansson, ...

© Björn Regnell, Lund University, 2015

<http://lth.se/programmera>

Innehåll

Om Kojo	1	Rita många polygoner	17	Spara djur i en vektor	35
Ditt första program	2	Värden och uttryck	18	Träna glosor	36
Rita en kvadrat	3	Sätt namn på värden med <code>val</code>	19	Huvudstadsspelet	37
Rita en trappa	4	Slumptal	20	Gör en timer med <code>object</code>	38
Gör en loop	5	Blanda dina egna färger	21	Simulera ett trafikljus	39
Rita en gubbe	6	Prova färgväljaren	22	Styr paddan med tangentbordet	40
Hur snabb är din dator?	8	Rita slumpcirklar	23	Styr paddan med musen	41
Spåra programmet	9	Rita en blomma	24	Gör ett ditt eget bankkonto	42
Gör din egen funktion med <code>def</code>	10	Skapa en variabel med <code>var</code>	25	Gör många objekt från en <code>class</code>	43
Stapla kvadrater	11	Rita många blommor	26	Prata med datorn	44
Gör en stapelfunktion	12	Byt kostym på paddan	27	Modda pong-spelet	45
Gör ett rutnät	13	Gör många paddor med <code>new</code>	28		
Kvadrat med parameter	14	Gör en kapplöpning	29		
Rita en kvadratgubbe	15	Alternativ med <code>if</code>	30		
Rita en polygon	16	Reagera på vad användaren gör	31		
		Gör en <code>while</code> -loop	32		
		Gissa talet	33		
		Träna multiplikation	34		

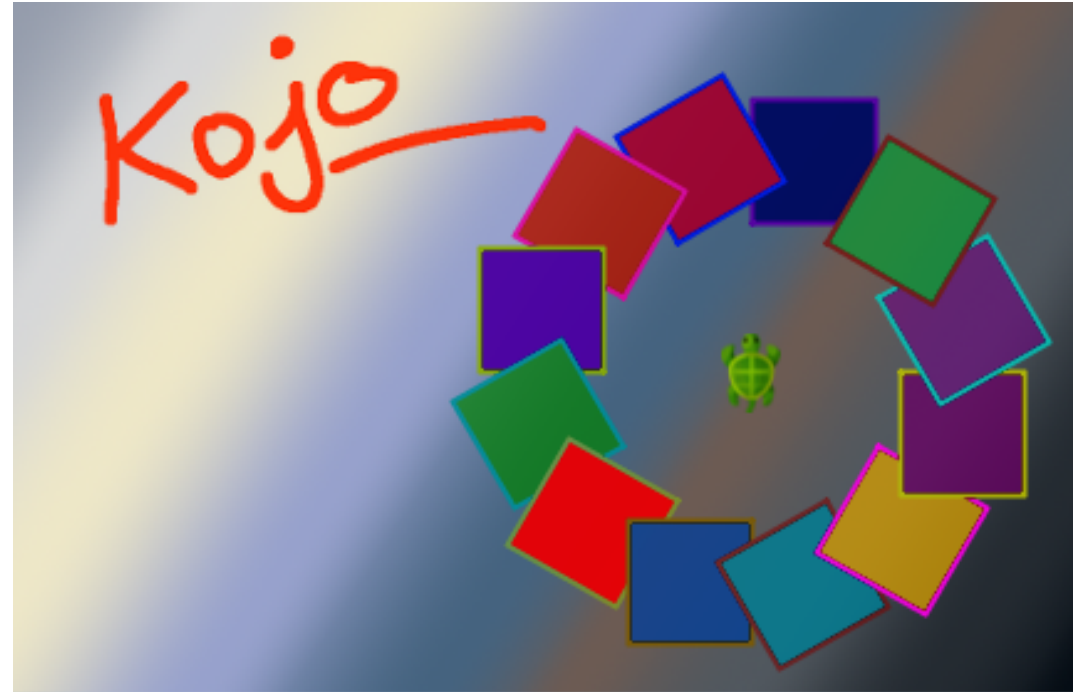
Om Kojo

Vad är Kojo?

Kojo är en app som hjälper dig att lära dig att programmera. Med Kojo kan du koda i det moderna och kraftfulla programspråket **Scala**. Kojo är gratis och finns på Svenska. Kojo fungerar med Linux, Windows och Mac OSX.

Var hittar jag Kojo?

Ladda ner Kojo här:
www.kogics.net/kojo-download
Läs mer här:
lth.se/programmera



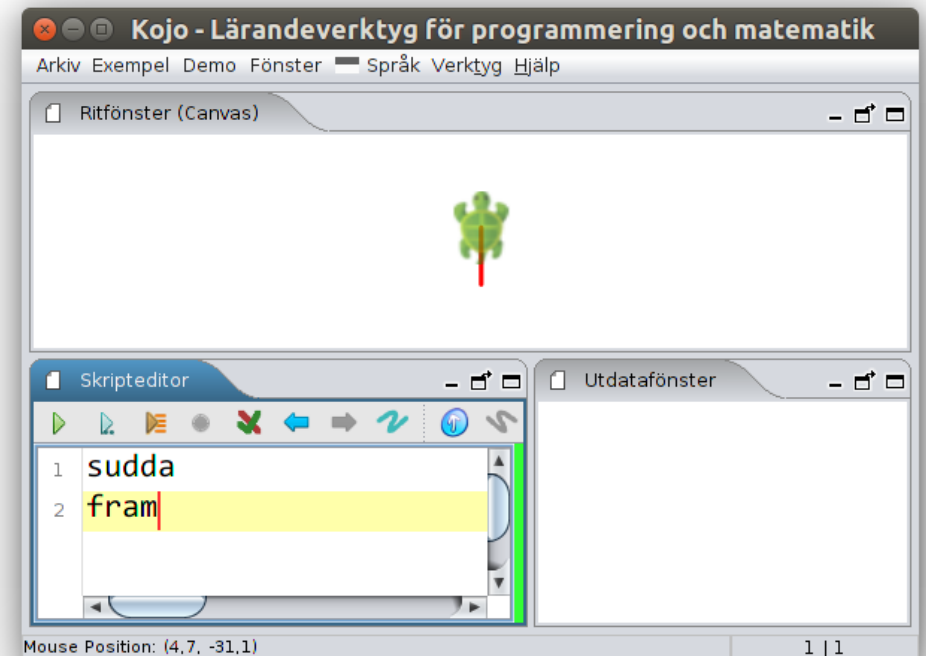
Ditt första program

Uppdrag:

Skriv så här i Kojos skripteditor-fönster:

```
sudda  
fram
```

Tryck på den gröna play-knappen
för att köra igång ditt program.



Rita en kvadrat

sudda
fram
höger

Om du skriver vänster eller höger så vrider sig paddan.

Uppdrag:

Utöka programmet så att det blir en kvadrat.



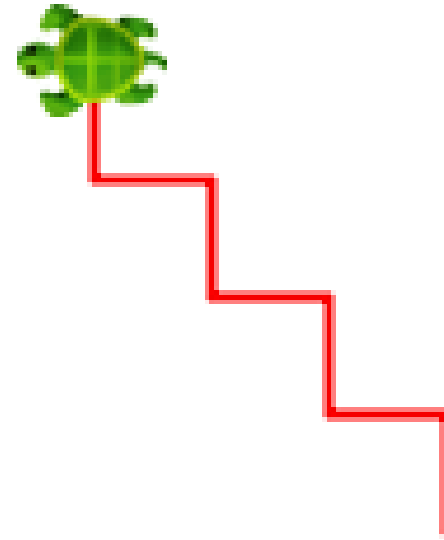
Rita en trappa

```
sudda  
fram; vänster  
fram; höger
```

Med semikolon ; mellan satserna kan du ha flera satser på samma rad.

Uppdrag:

Utöka programmet så att det blir en trappa.



Gör en loop

sudda

```
upprepa(4){ fram; höger }
```

Uppdrag:

- Vad händer om du ändrar 4 till 100?
- Rita en trappa med 100 trappsteg.



Rita en gubbe

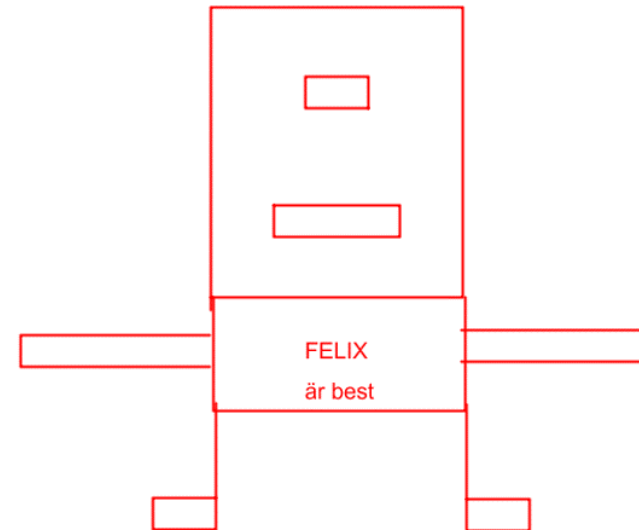
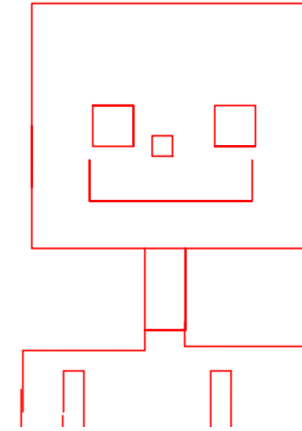
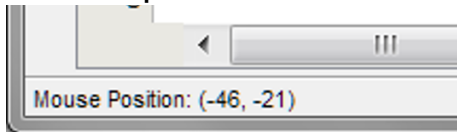
Uppdrag:

Rita en gubbe som du själv vill.

Tips:

```
hoppa  
vänster(180)  
fram(300)  
hoppa(100)  
hoppaTill(25,-28)  
skriv("FELIX är bäst")  
färg(lila)  
fyll(grön)
```

Du kan se paddans läge nere till vänster medan du rör muspekaren i Ritfönstret:



Hur snabb är din dator?

Den första elektroniska datorn hette **ENIAC** och kunde räkna till 5000 på en sekund.
I Kojo finns en funktion `räknaTill` som mäter hur snabbt datorn kan räkna.
När jag kör `räknaTill(5000)` på min snabba dator skrivs detta i utdata-fönstret:

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

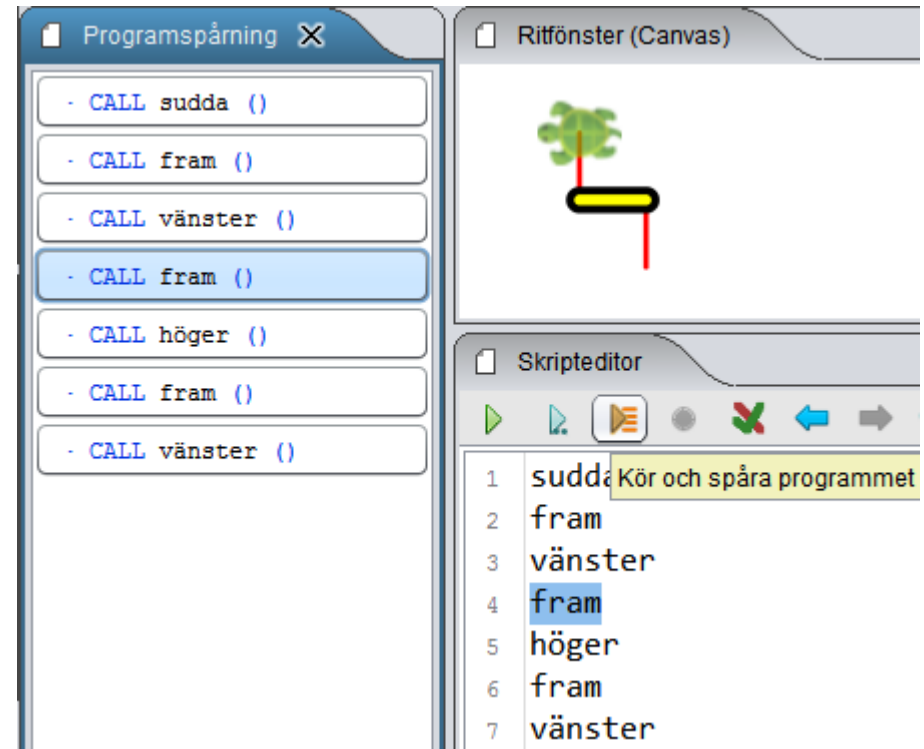
Uppdrag:

- Kör `räknaTill(5000)` och kolla om din dator är snabbare än min.
- Hur lång tid tar det för din dator att räkna till en miljon?
- Hur långt hinner din dator räkna till på en sekund?

Spåra programmet

Uppdrag:

- Skriv ett program som ritar ett trappsteg.
- Tryck på den orange-färgade play-knappen.
- Klicka på ett av anropen: CALL fram. Vad händer i Ritfönstret?
- När en del av programmet är markerad med blått körs bara denna del om du trycker play. Avmarkera genom att klicka bredvid markeringen.
- Lägg till fler satser i ditt program och se vad som händer när du spårar.
- Stäng fönstret *Programspårning* när du är klar.



Gör din egen funktion med **def**

Med **def** kan du göra egna *funktioner* som du själv väljer namn på.

```
def kvadrat = upprepa(4){ fram; höger }
```

sudda

```
kvadrat //använd din kvadrat-funktion
```

hoppa

```
kvadrat
```

Uppdrag:

- Byt färg på kvadraterna.
- Gör fler kvadrater.

Tips:

```
fyll(grön); färg(lila)
```

Stapla kvadrater

Uppdrag:

Gör en stapel med 10 kvadrater.

Tips:

```
def kvadrat = upprepa(4){ fram; höger }
```

```
sudda; sakta(100)  
upprepa(10){ ??? }
```



Gör en stapelfunktion

Uppdrag:

Gör en funktion som heter `stapel`, som ritar en stapel med 10 kvadrater.

Tips:

```
def kvadrat = upprepa(4){ fram; höger }  
def stapel = ???
```

```
sudda; sakta(100)  
stapel
```



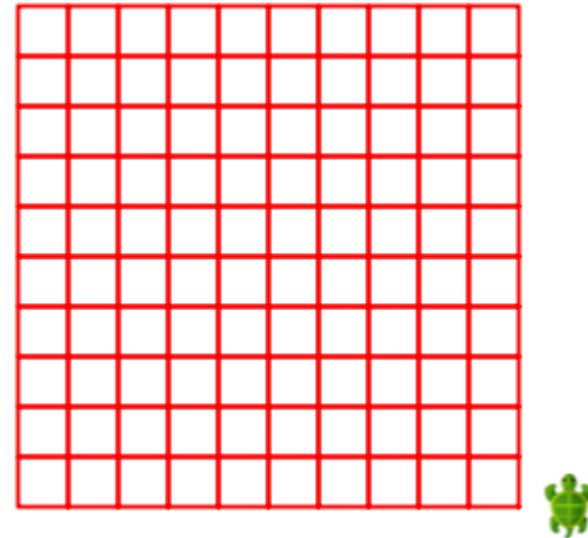
Gör ett rutnät

Uppdrag:

Gör ett rutnät med 10*10 kvadrater.

Tips:

- Använd din stapelfunktion från tidigare.
- Du kan hoppa baklänges en hel stapelhöjd med `hoppa(-10 * 25)`
- Du kan sedan hoppa till rätt plats med `höger; hoppa; vänster`



Kvadrat med parameter

Uppdrag:

Rita olika stora kvadrater.

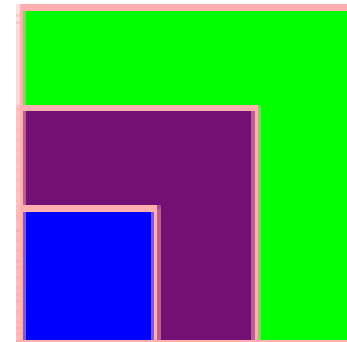
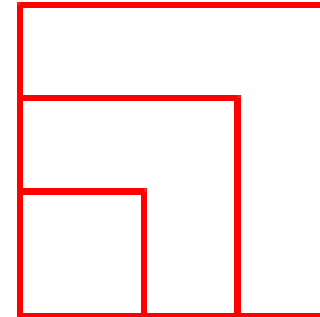
Tips:

Ge din kvadrat-funktion en *parameter*,
med namnet *sidlängd* och typen *Heltal*:

```
def kvadrat(sidlängd : Heltal) =  
  upprepa(4){ fram(sidlängd); höger }
```

```
sudda; sakta(100); osynlig  
kvadrat(100)  
kvadrat(70)  
kvadrat(40)
```

Du kan byta färg med:
fyll(blå); färg(rosa)



Rita en kvadratgubbe

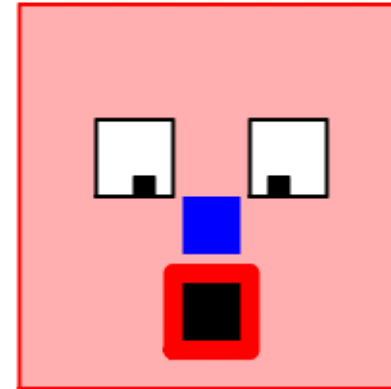
Uppdrag:

Rita en gubbe med hjälp av olika stora kvadrater.

Tips:

```
def kvadrat(x: Heltal, y: Heltal, sidlängd: Heltal) = {  
  hoppaTill(x, y)  
  upprepa(4) { fram(sidlängd); höger }  
}  
def huvud(x: Heltal, y: Heltal) = { fyll(rosa); färg(röd); kvadrat(x, y, 200) }  
def öga(x: Heltal, y: Heltal) = { fyll(vit); färg(svart); kvadrat(x, y, 40) }  
def pupill(x: Heltal, y: Heltal) = { fyll(svart); färg(svart); kvadrat(x, y, 10) }  
def näsa(x: Heltal, y: Heltal) = { fyll(blå); färg(genomskinlig); kvadrat(x, y, 30) }  
def mun(x: Heltal, y: Heltal) = { bredd(10); fyll(svart); färg(röd); kvadrat(x, y, 40) }
```

```
sudda; sakta(20); osynlig  
huvud(0, 0)  
öga(40, 100); pupill(60, 100)  
???
```



Rita en polygon

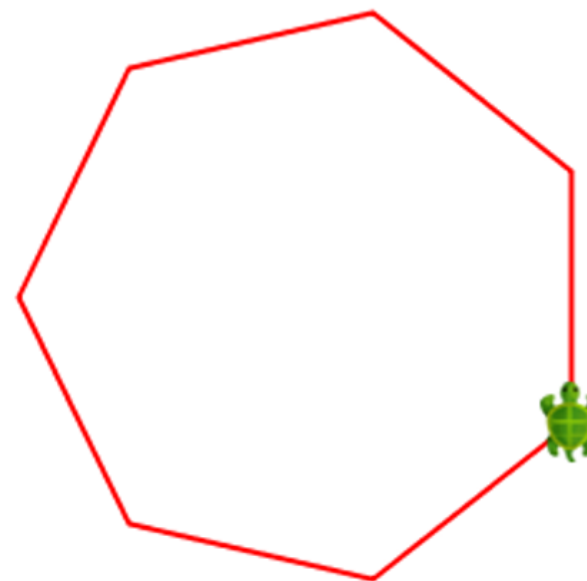
Uppdrag:

- Prova koden nedan. Rita olika slags polygoner.
- Lägg till en parameter sidlängd och rita olika stora polygoner.
- Hur stort behöver n vara för att det ska se ut som en cirkel?

Tips:

```
def polygon(n:Heltal) = upprepa(n){  
  fram(100)  
  vänster(360.0/n)  
}
```

```
sudda; sakta(100)  
polygon(7)
```

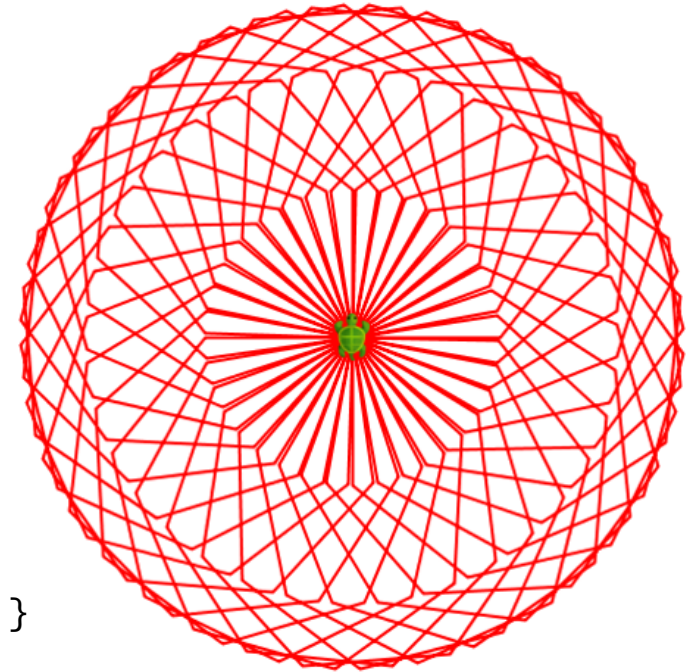


Rita många polygoner

Uppdrag:

- Prova programmet nedan.
- Prova ändra antalet sidor och vinkel.
- Fyll polygonerna med färg.

```
def polygon(n: Heltal, sidlängd: Heltal) = upprepa(n){  
  fram(sidlängd)  
  vänster(360.0/n)  
}  
def snurra(n: Heltal, vinkel: Heltal, sidlängd: Heltal) =  
  upprepa(360/vinkel){ polygon(n, sidlängd); vänster(vinkel) }  
  
sudda; sakta(5)  
snurra(7, 10, 100)
```



Värden och uttryck

Uppdrag:

- Skriv `1 + 1` och tryck på den blå play-knappen. Då skapar kojo en grön kommentar.
- Kommentaren visar att värdet av uttrycket `1 + 1` är 2 och att typen är `Int`, som betyder Heltal.
- Gör fler uträkningar. Vad det blir för värde och typ?

`5 * 5`

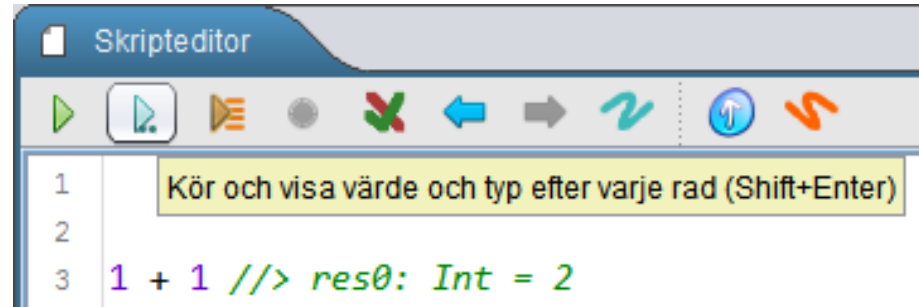
`10 + 2 * 5`

`"hej" + "på" + "dej"`

`5 / 2`

`5 / 2.0`

`5 % 2`



Tips:

- Med `/` mellan heltal blir det heltalsdivision och decimalerna kastas bort. För att det ska bli division med decimaler måste minst ett av talen vara ett decimaltal.
- Med `%` får du resten vid en heltalsdivision.

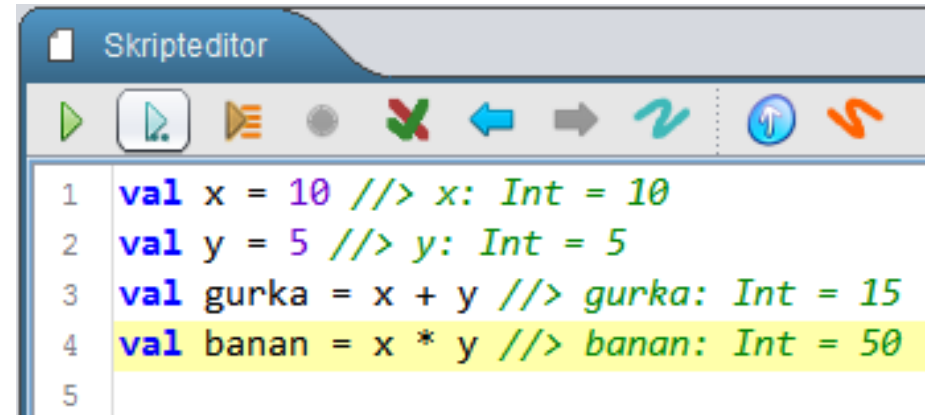
Sätt namn på värden med **val**

Uppdrag:

Med **val** kan du koppla ett namn till ett värde. Namnet kan sedan användas istället för värdet. Prova programmet nedan. Vad skriver paddan?

```
val x = 10  
val y = 5  
val gurka = x + y  
val banan = x * y
```

```
sudda  
fram; skriv(banan)  
fram; skriv(gurka)  
fram; skriv(y)  
fram; skriv(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
```

Slumptal

Uppdrag:

- Kör programmet nedan många gånger. Vad händer?
- Vilket är det minsta och största möjliga värdet på radien r ?
- Ändra så att r blir ett slumptal mellan 3 och 200.
- Rita 100 cirklar med slumpmässig radie på slumpmässig plats, som bilden visar.

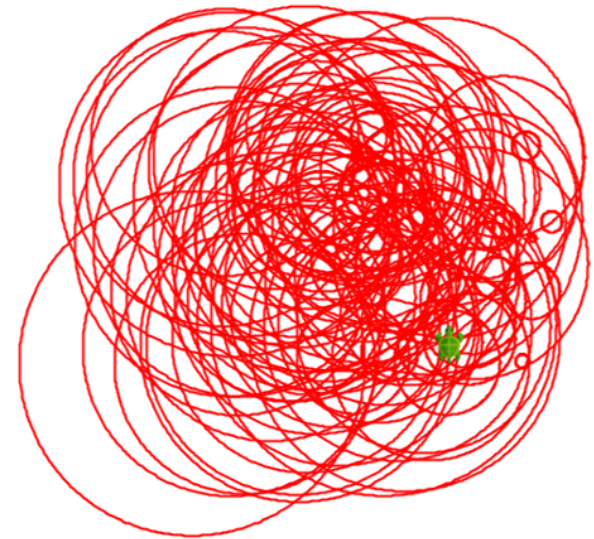
```
//värdet r blir ett slumptal mellan 10 och 89:
```

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

```
sudda; sakta(10); osynlig
```

```
skriv("Radie = " + r)
```

```
cirkel(r)
```



Blanda dina egna färger

- Med Color kan du blanda egna färger, till exempel Color(0, 70, 0)
- De tre parametrarna anger mängden *rött*, *grönt* och *blått*
- Du kan också lägga till en fjärde parameter som anger *genomskinligheten*
- Alla parametrar ska vara mellan 0 och 255

Uppdrag:

Prova programmet nedan. Ändra genomskinligheten.

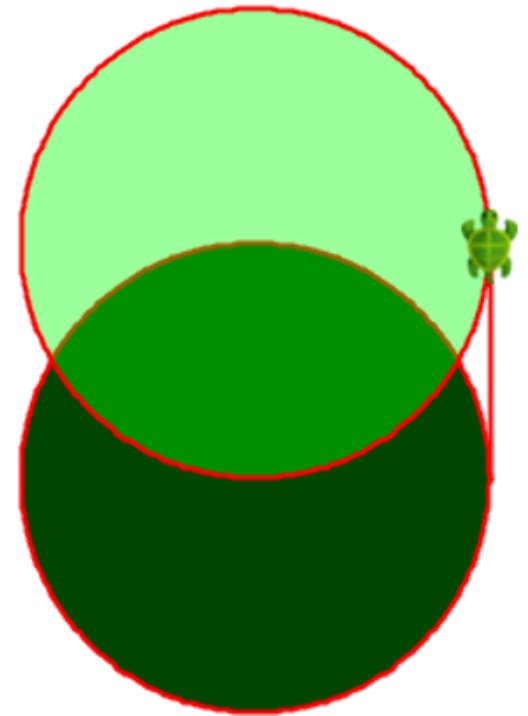
```
sudda; sakta(100)
```

```
val olivgrön = Color(0,70,0)
```

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

```
fyll(olivgrön); cirkel(100)
```

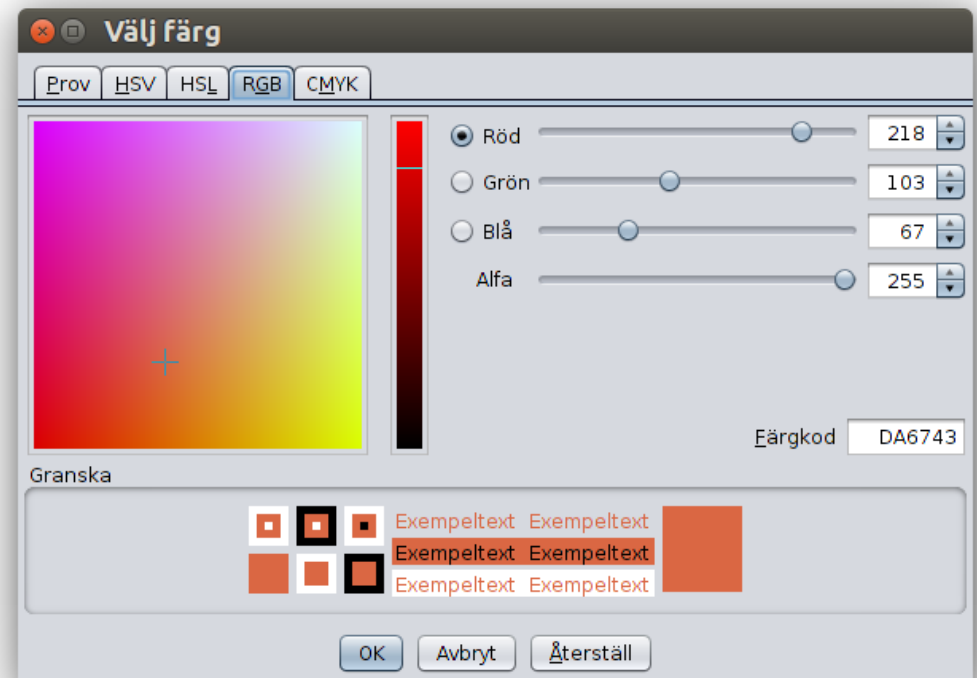
```
fyll(pistageglass); fram(100); cirkel(100)
```



Prova färgväljaren

Uppdrag:

- Högerklicka i editor-fönstret och klicka på Välj färg...
- Om du väljer fliken **RGB** i färgväljaren kan du blanda nya RGB-färger.
- Tryck OK och titta i Utdatafönstret. Där syns de tre RGB-värdena för rött, grönt och blått.
- Du kan använda dessa värden i ditt program för att rita med din nya färg med `färg(Color(218, 153, 67))`.



Rita slumpcirkclar

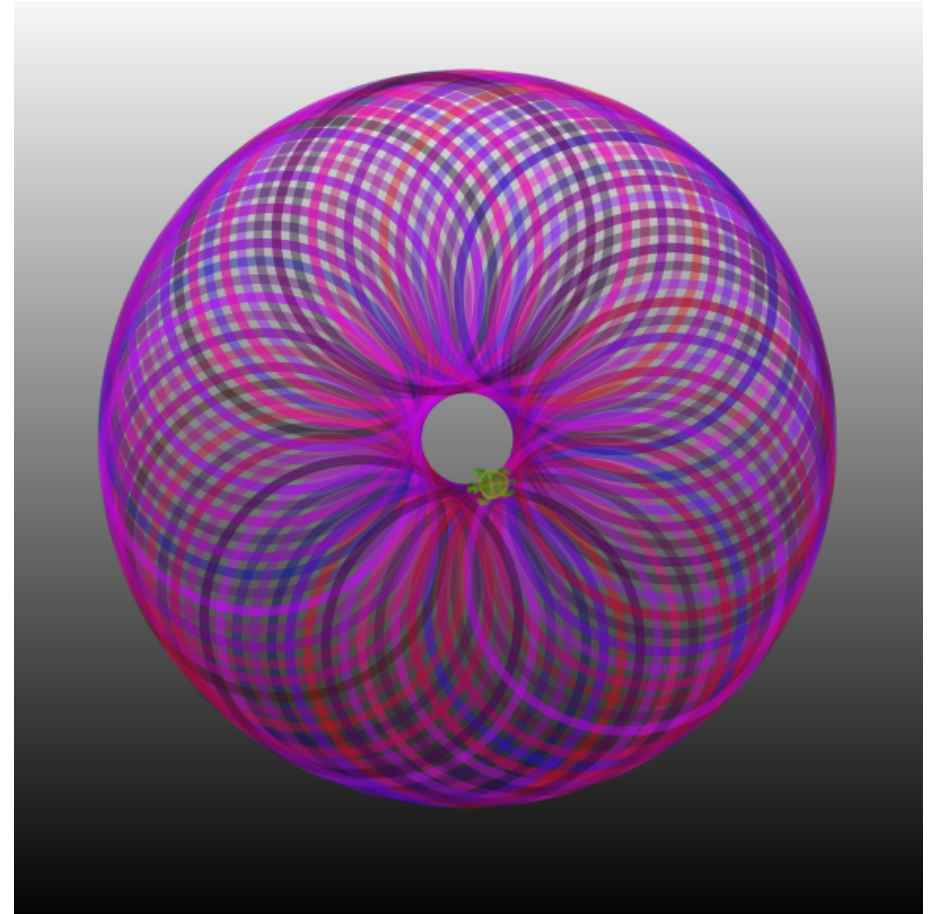
```
def slump = slumptal(256)
def slumpfärg = Color(slump,10,slump,100)

sudda; sakta(5)
bakgrund2(svart,vit)
bredd(6)

upprepa(100) {
  färg(slumpfärg)
  cirkel(100)
  hoppa(20)
  höger(35)
}
```

Uppdrag:

Prova olika slumpfärger och bakgrunder.

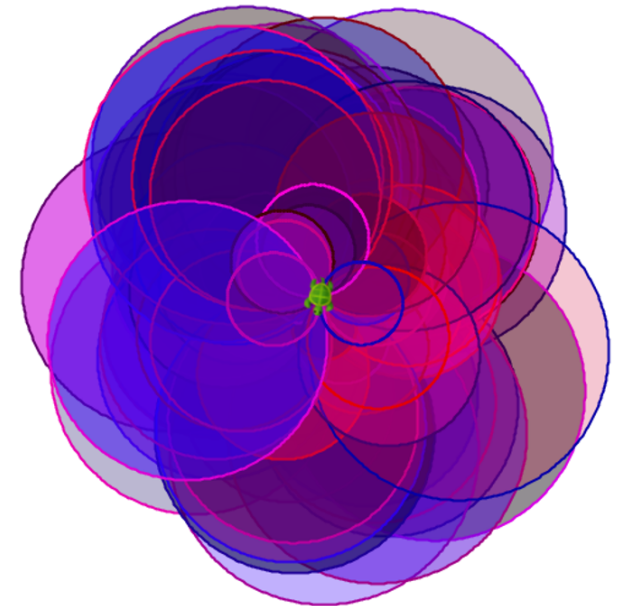


Rita en blomma

Uppdrag:

Programmet nedan ritar 100 slumpfärgade cirklar på slumpmässig plats med slumpmässig radie. Prova att ändra de olika slumptalens gränser och försök förklara vad som händer.

```
sudda(); sakta(5)
bredd(2)
upprepa(100){
  färg(Color(slumptal(256),0,slumptal(256)))
  fyll(Color(slumptal(256),0,slumptal(256),slumptal(100)+50))
  vänster(slumptal(360))
  cirkel(slumptal(30)*4+10)
}
```



Skapa en variabel med **var**

Med **var** kan koppla ett namn till ett värde.

Du får då en variabel, som kan tilldelas ett nytt värde så här:

```
var gurka = 1  
gurka = 1 + 1 //först räknas 1 + 1 ut, sedan blir gurka 2
```

Uppdrag:

Prova programmet nedan. Vad skriver paddan?

```
var i = 0  
  
sudda  
upprepa(10){  
  i = i + 1  
  fram; skriv(i)  
}
```

Tips:

- I satsen `i = i + 1` tilldelas `i` ett nytt värde som blir det *gamla* värdet av `i` plus 1

Rita många blommor

Uppdrag:

- Gör en funktion som heter blomma, som ritar en krona och en stjälk från kronans mitt med ett grönt blad.
- Rita 5 blommor bredvid varandra.

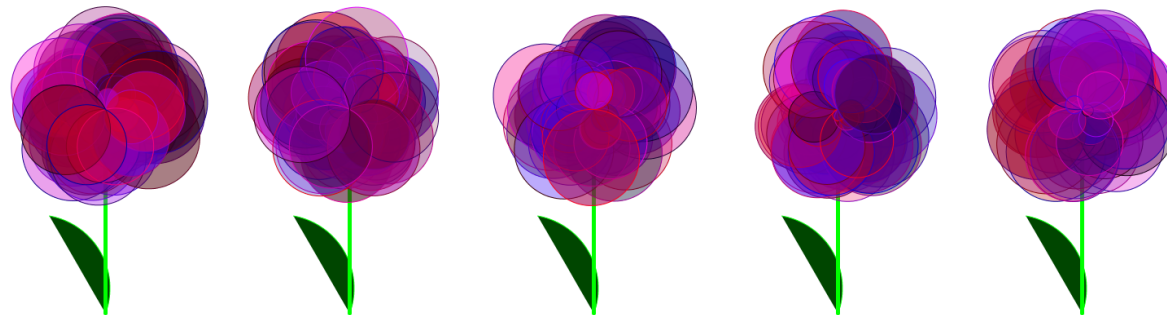
Tips:

Du kan rita blad med `båge(radie, vinkel)`.

Låt funktionen `blomma` ha två parametrar `x` och `y` och använd `hoppaTill(x,y)`

Du kan loopa 5 gånger och räkna ut platsen så här:

```
var i = 0
upprepa(5){
  blomma(600*i,0)
  i = i + 1
}
```



Byt kostym på paddan

Uppdrag:

Ladda ner mediafiler från Kojos hemsida: www.kogics.net/kojo-download#media

- Packa upp filen `scratch-media.zip` och leta rätt på krabbbilden `crab1-b.png` i mappen `Media/Costumes/Animals`
- Lägg filen `crab1-b.png` i samma mapp som du har ditt program.
- Prova att byta kostym på paddan till en krabba så här:

```
sudda  
kostym("crab1-b.png")  
sakta(2000)  
fram(1000)
```



Tips:

- Du kan också använda dina egna bilder av typen `.png` eller `.jpg`
- Om du vill lägga bilden i en annan mapp så kan du skriva filens sökväg, till exempel `kostym("~/Kujo/Media/Costumes/Animals/crab1-b.png")` där `~` betyder din hemkatalog.

Gör många paddor med **new**

Du kan skapa många nya paddor med **new** så här:

sudda

```
val p1 = new Padda(100,100) //nya paddan p1 börjar på plats (100, 100)
```

```
val p2 = new Padda(100, 50) //nya paddan p2 börjar på plats (100, 50)
```

```
p1.fram(100)
```

```
p2.fram(-100) //paddan p2 backar
```

Uppdrag:

- Skapa tre paddor som står ovanför varandra.
- Gör så att alla huvud är vända åt vänster.

Tips:

- p1 och p2 är paddornas *namn*. Du kan välja vilka namn du vill.
- Med namnet p1 och en punkt kan du ge instruktioner till paddan p1 så här: p1.vänster
- Med `osynlig` blir den vanliga paddan osynlig.



Gör en kapplöpning

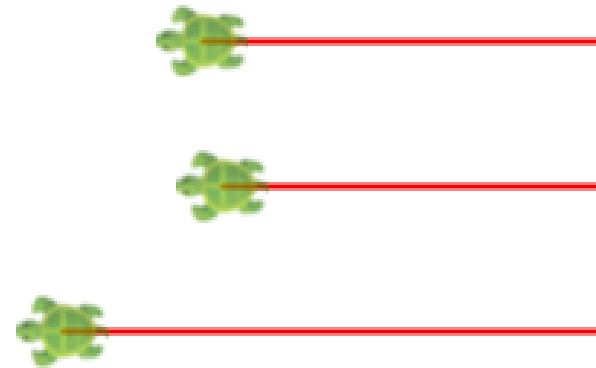
Med hjälp av slumpal kan paddorna genomföra en kapplöpning mot varandra.

Uppdrag:

- Låt tre paddor springa ikapp.
- Om alla får springa fram 10 gånger, vilken padda kommer då först?

Tips:

- Med `p1.fram(slumptal(100) + 1)` går paddan p1 fram 1 till 100 steg.



Alternativ med **if**

Med en **if**-sats kan datorn välja mellan två olika alternativ.

sudda; osynlig

```
if (true) skriv("sant") else skriv("falskt")
```

Uppdrag:

- Ändra **true** till **false** och kolla vad paddan skriver.
- Ändra villkoret till $2 > 1$ och kolla vad paddan skriver.
- Ändra villkoret till $2 < 1$ och kolla vad paddan skriver.
- Förklara hur en **if**-sats fungerar.

Tips:

- Om villkoret efter **if** är **true** väljs det som står efter villkoret.
- Om villkoret efter **if** är **false** väljs det som står efter **else**.

Reagera på vad användaren gör

```
suddaUtdata; setOutputTextFontSize(35)
val lösenord = "gurka"
val fråga    = "Vad är lösenordet?"
val rätt     = "Kassaskåpet är öppet!"
val fel      = "Du får inte komma in!"
val svar = indata(fråga) //vänta på svar från användaren
val meddelande = if (svar == lösenord) rätt else fel
utdata(meddelande)
```

Uppdrag:

- Prova programmet och förklara vad som händer.
- Ändra lösenord, fråga och vad som skrivs ut när det blev rätt ocv fel.
- Fråga även efter användarnamn och lägg till användarnamnet i utskriften.

Gör en **while**-loop

Med en **while**-loop kan datorn upprepa satser så länge ett villkor är sant.

```
sudda; osynlig; sakta(250); suddaUtdata
var x = 200
while (x > 0) { //kolla villkoret före varje runda
    fram(x); höger
    skriv(x)
    x = x - 12
}
utdata("x är nu: " + x)
```

Uppdrag:

- Vad skrivs ut i utdatafönstret? Varför?
- Spåra programmet med den orange-färgade play-knappen och undersök varje steg.
- Ändra minskningen av x från 12 till 20. Förklara vad som händer.

Gissa talet

```
val hemlis = slumpTal(100)+1
var svar = indata("Gissa ett tal mellan 1 och 100! ")
var fortsätt = true

while (fortsätt) {
    if (svar.toInt < hemlis)
        svar = indata(svar + " är för LITET, gissa igen!")
    else if (svar.toInt > hemlis)
        svar = indata(svar + " är för STORT, gissa igen!")
    else if (svar.toInt == hemlis)
        fortsätt = false
}
utdata(hemlis + " är RÄTT svar!")
```

Uppdrag:

Inför en variabel `var antalFörsök = 0` och se till att utskriften på slutet blir:
Rätt svar! Du klarade det på 5 gissningar

Träna multiplikation

```
var antalRätt = 0
val startTid = System.currentTimeMillis / 1000
upprepa(12) {
    val tal1 = slumpTal(12)+1
    val tal2 = slumpTal(12)+1
    val svar = indata("Vad är " + tal1 + "*" + tal2 + "?")
    if (svar == (tal1 * tal2).toString) {
        utdata("Rätt!")
        antalRätt = antalRätt + 1
    }
    else utdata("Fel. Rätt svar är " + (tal1 * tal2))
}
val stoppTid = System.currentTimeMillis / 1000
val sek = stoppTid - startTid
utdata("Du fick " + antalRätt + " rätt på " + sek + " sekunder.")
```

Uppdrag:

Ändra så att man bara tränar 8:ans och 9:ans tabell.

Spara djur i en vektor

```
var djur = Vector("älg", "ko", "kanin", "kvalster") //variablen djur blir en vektor med 4 djur
utdata("Första djuret i vektorn är: " + djur(0)) //platserna i vektorer räknas från 0
utdata("Andra djuret i vektorn är: " + djur(1))
utdata("Det finns så här många djur: " + djur.size)
utdata("Sista djuret i vektorn är: " + djur(djur.size-1))
```

```
val s = slumptal(djur.size) //dra ett slumpal mellan 0 och antalet djur minus 1
utdata("Ett slumpmässigt djur: " + djur(s))
```

```
djur = djur :+ "kamel" //lägg till ett djur sist i vektorn
djur = "dromedar" +: djur //lägg till ett djur först i vektorn
djur = djur.updated(2, "slamkrypare") //Ändra tredje djuret (plats 2 i vektorn)
utdata("Alla djur i vektorn baklänges:")
djur.foreach{ x => utdata(x.reverse) } //för alla x i vektorn: skriv ut x baklänges
```

Uppdrag:

- Vad skriver programmet i utdatafönstret? Förklara vad som händer.
- Lägg till fler djur i vektorn.

Träna glosor

```
val svenska = Vector("dator", "sköldpadda", "cirkel")
val engelska = Vector("computer", "turtle", "circle")
var antalRätt = 0
upprepa(5) {
    val s = slumptal(3)
    val glosa = svenska(s)
    val svar = indata("Vad heter " + glosa + " på engelska?")
    if (svar == engelska(s)) {
        utdata("Rätt svar!")
        antalRätt = antalRätt + 1
    } else {
        utdata("Fel svar. Rätt svar är: " + engelska(s))
    }
}
utdata("Du fick " + antalRätt + " rätt.")
```

Uppdrag:

- Lägg till fler glosor.
- Träna på glosor från engelska till svenska.
- Låt användaren välja hur många frågor innan avslut. Tips:
`val antal = indata("Ange antal: ").toInt`

Huvudstadsspelet

```
def huvudstadsspelet = {  
  println("Välkommen till Huvudstadsspelet!")  
  val stad = Map("Sverige" -> "Stockholm", "Danmark" -> "Köpenhamn", "Skåne" -> "Malmö")  
  var länderKvar = stad.keySet //keySet ger en mängd av alla nycklar i en Map  
  def slumpLand = scala.util.Random.shuffle(länderKvar.toVector).head  
  while(!länderKvar.isEmpty) {  
    val land = slumpLand  
    val svar = indata("Vad heter huvudstaden i " + land + "?")  
    utdata(s"Du skrev: $svar")  
    if (svar == stad(land)) {  
      utdata("Rätt svar! Du har " + länderKvar.size + " länder kvar!")  
      länderKvar = länderKvar - land //ta bort land ur mängden länderKvar  
    } else utdata(s"Fel svar. Huvudstaden i $land börjar på ${stad(land).take(2)}...")  
  }  
  utdata("TACK FÖR ATT DU KÄMPADE! (Tryck ESC)")  
}  
  
toggleFullScreenOutput;  
setOutputBackground(black); setOutputTextColor(green); setOutputTextFontSize(30)  
upprepa(100)(utdata("")) //scrolla utdafönstret med 100 blanka rader  
huvudstadsspelet  
  
// *** UPPDRAG: (1) Lägg till fler par: land -> stad (2) Mät tid och räkna poäng.
```

Gör en timer med **object**

```
object timer {  
  def nu = System.currentTimeMillis //ger nutid i millisekunder  
  var tid = nu  
  def nollställ = { tid = nu }  
  def mät = nu - tid  
  def slumpvänta(min: Int, max: Int) = //vänta mellan min och max sekunder  
    Thread.sleep((slumptal(max-min)+min)*1000) //Thread.sleep(1000) väntar 1 sekund  
}  
  
utdata("Klicka i utdatafönstret och vänta...")  
timer.slumpvänta(3,6) //vänta mellan 3 och 6 sekunder  
timer.nollställ  
indata("Tryck Enter så snabbt du kan.")  
utdata("Reaktionstid: " + (timer.mät/1000.0) + " sekunder")
```

Med **object** kan du samla saker som hör ihop i ett objekt.

Du kommer åt en sak inne i ett objekt med en punkt: `timer.nollställ`

Uppdrag:

- Prova programmet och mät din reaktionstid. Hur snabb är du?
- Använd timer i uppgiften *Gissa talet* och lägg till utskriften:
Rätt svar! Du klarade det på 5 gissningar och 32 sekunder

Simulera ett trafikljus

```
def släckAlla = draw(penColor(gray) * fillColor(black) -> PicShape.rect(130,40))
def ljus(c: Color, h: Int) = penColor(noColor) * fillColor(c) * trans(20,h) -> PicShape.circle(15)
def tändRött = draw(ljus(red, 100))
def tändGult = draw(ljus(yellow, 65))
def tändGrönt = draw(ljus(green, 30))
def vänta(sekunder: Int) = Thread.sleep(sekunder*1000)

sudda; osynlig
while (true) { //en oändlig loop
  släckAlla
  tändRött; vänta(3)
  tändGult; vänta(1)
  släckAlla
  tändGrönt; vänta(3)
  tändGult; vänta(1)
}
```



Uppdrag:

- Hur växlar trafikljuset? Försök förklara vad som händer.
- Ändra så att trafikljuset är grönt dubbelt så länge.

Styr paddan med tangentbordet

```
sudda; sakta(0)
activateCanvas()

animate { fram(1) }

onKeyPress { k =>
  k match {
    case Kc.VK_LEFT => vänster(5)
    case Kc.VK_RIGHT => höger(5)
    case Kc.VK_SPACE => fram(5)
    case _ =>
      utdata("Annan tangent: " + k)
  }
}
```

Uppdrag:

- Skriv Kc. och tryck Ctrl+Alt+Mellanslag och kolla vad de olika tangenterna heter.
- Gör pennaUpp om man trycker pil upp
- Gör pennaNer om man trycker pil ner
- Gör färg(blå) om man trycker B
- Gör färg(röd) om man trycker R
- Öka eller minska hastigheten om man trycker + eller -

Styr paddan med musen

```
sudda; sakta(100)
activateCanvas()
```

```
var rita = true
```

```
onKeyPress { k =>
  k match {
    case Kc.VK_DOWN =>
      penDown()
      rita = true
    case Kc.VK_UP =>
      penUp()
      rita = false
    case _ =>
      utdata("Annan tangent: " + k)
  }
}

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

Uppdrag:

- Gör `fill(svart)` om man trycker på F
- Inför en variabel `var fyllNästa = true` och i fallet att man trycker på `Kc.VK_F` gör:

```
if (fyllNästa) {
  fill(svart)
  fyllNästa=false
} else {
  fill(genomskinlig)
  fyllNästa=true
}
```

Gör ett ditt eget bankkonto

```
object mittKonto {  
  val nummer = 123456  
  var saldo = 0.0  
  def in(belopp: Decimaltal) = {  
    saldo = saldo + belopp  
  }  
  def ut(belopp: Decimaltal) = {  
    saldo = saldo - belopp  
  }  
  def visaSaldo() = {  
    utdata("Konto nummer: " + nummer)  
    utdata("      saldo: " + saldo)  
  }  
}
```

```
mittKonto.visaSaldo()  
mittKonto.in(100)  
mittKonto.visaSaldo()  
mittKonto.ut(10)  
mittKonto.visaSaldo()
```

Uppdrag:

- Vad är saldot efter att programmet kört klart? Förklara vad som händer.
- Gör så att det inte går att ta ut mer pengar än som finns på kontot.
- Lägg till **val** maxBelopp = 5000 och kolla så att man inte kan ta ut mer än maxBelopp åt gången.

Gör många objekt från en **class**

Om man vill skapa många konto behövs en klass. Med **new** skapas nya objekt. Varje objekt får eget nummer och saldo.

```
class Konto(nummer: Heltal) {  
  private var saldo = 0.0 //private betyder "hemlig"  
  def in(belopp: Decimaltal) = {  
    saldo = saldo + belopp  
  }  
  def ut(belopp: Decimaltal) = {  
    saldo = saldo - belopp  
  }  
  def visaSaldo() =  
    utdata(s"Konto $nummer: $saldo")  
}
```

```
val konto1 = new Konto(12345) //new skapar objekt  
val konto2 = new Konto(67890) //ännu ett objekt
```

```
konto1.in(99)  
konto2.in(88)  
konto1.ut(57)  
konto1.visaSaldo  
konto2.visaSaldo
```

Uppdrag:

- Vad är saldot på de olika kontona när programmet kört klart? Förklara vad som händer.
- Skapa ännu fler bankkonto-objekt och sätt in och ta ut lite pengar på dessa.
- Lägg till en klassparameter namn: String som ska innehålla namnet på kontoägaren när objekt skapas.
- Gör så att även namn skrivs ut när visaSaldo anropas
- Vad händer om du gör:
konto1.saldo = 10000000

Prata med datorn

```
setOutputBackground(black); setOutputTextFontSize(30); setOutputTextColor(green)
utdata("Skriv intressanta svar även om frågorna är konstiga. Avsluta med 'hej då'")
def slumpa(xs: Vector[String]) = scala.util.Random.shuffle(xs).head
val ledtexter = Vector("Vad betyder", "Gillar du", "Varför behövs", "Berätta mer om")
var svar = "?"
val öppning = "Vad vill du prata om?"
var ord = Vector("navelludd", "ketchupglass", "jultomten", "örngott")
while (svar != "hej då") {
  val t = if (svar == "?") öppning
    else if (svar == "nej") "Nähä."
    else if (svar == "ja") "Jaha."
    else if (svar.length < 4) "Jasså..."
    else slumpa(ledtexter) + " " + slumpa(ord) + "?"
  svar = indata(t).toLowerCase
  ord = ord ++ svar.split(" ").toList.filter(_.length > 3)
}
utdata("Tack för pratstunden! Jag kan nu dessa ord:" + ord)
```

//Uppdrag:

// (1) Prova programmet och försök att förklara vad som händer.

// (2) När avslutas while-loopen?

// (3) Lägg till fler strängar i vektorerna ledtexter och ord

// (4) Lägg till fler bra svar på några korta ord utöver "nej" och "ja"

Modda pong-spelet

Uppdrag:

- Välj menyn Exempel > Animeringar och spel > Pong och prova spelet.
- Man styr med pil upp och pil ner, samt A och Z.
- Tryck ESC för att avbryta spelet och undersök koden.
- Ändra i koden så att bollen blir större.
- Gör spelplanen till en tennisplan, med grönt underlag, vita linjer och en gul boll.

