Velkommen

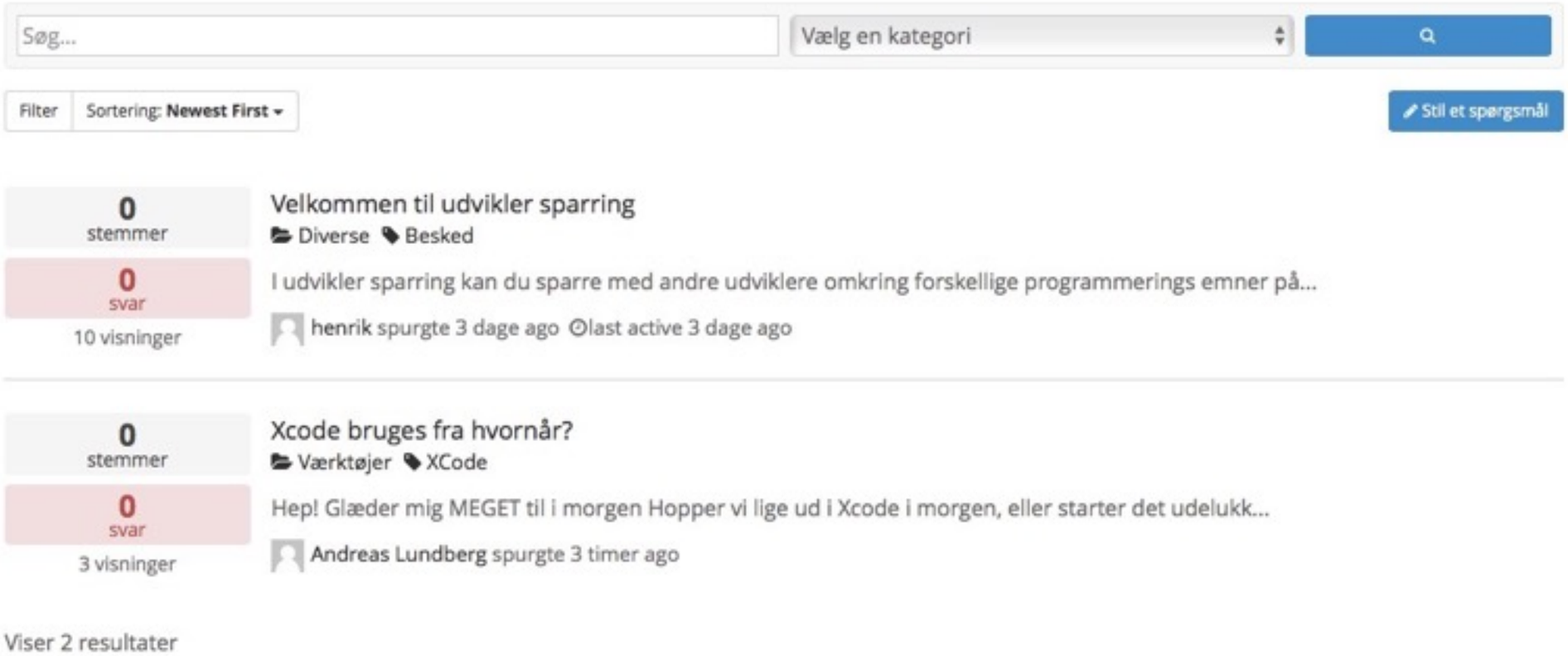
Lektion 1

Fundamentet

Agenda

- Velkommen, god tone og orden.
- Udvikler sparring.
- De 3 grund trin.
- Computerens DNA (Bit).
- Hukommelsen (RAM).
- Sådan fungerer et udviklingsmiljø
- De basale typer i Swift.
- Typecasting.
- Stak.
- Array.
- Øvelser

God tone



Udvikler sparring

De 3 grund trin

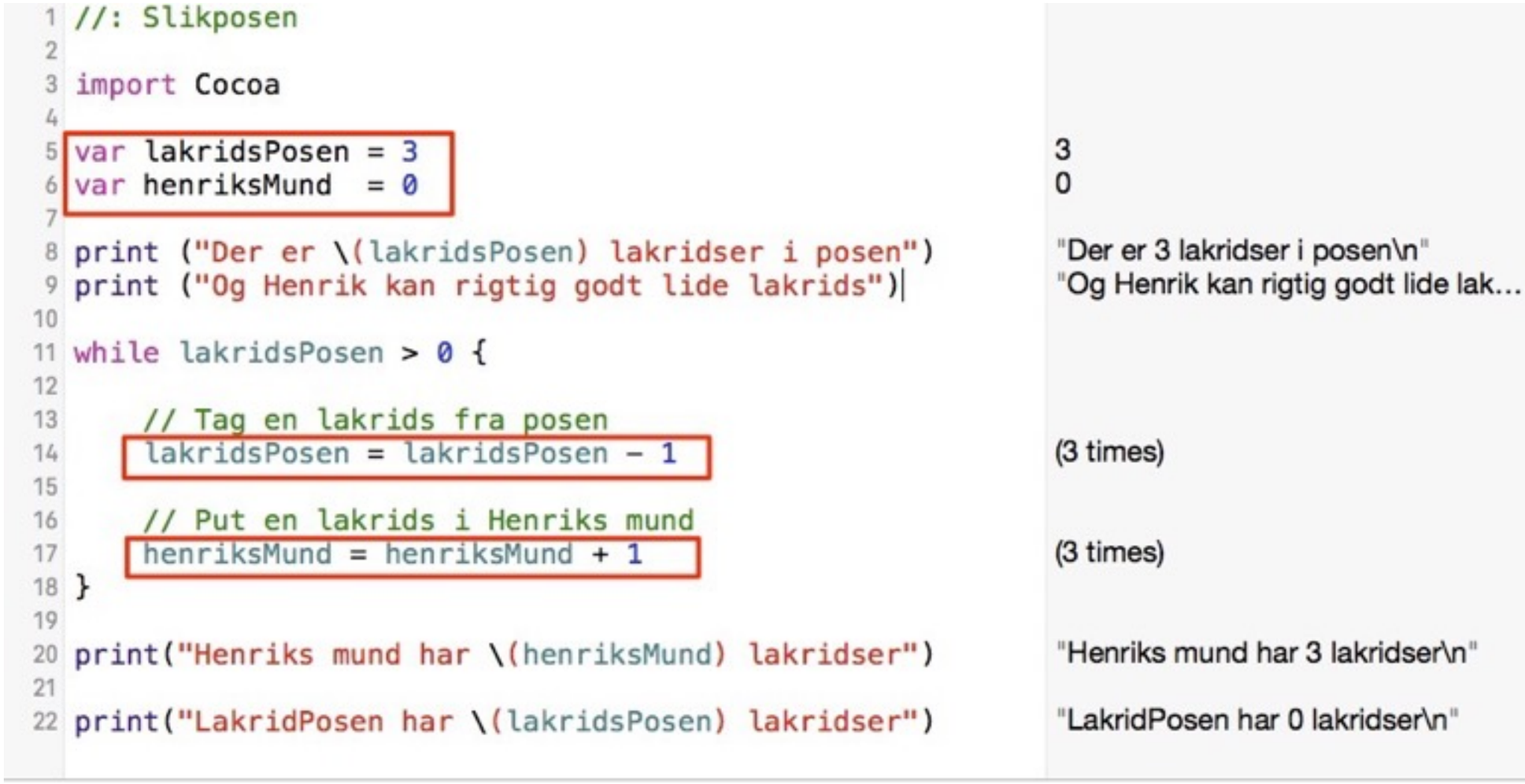
- Tildeling (Asignment)
- Sammenligning (Selektion)
- Gentagelse (Iteration)

```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Eksempel

I den gode sags tjeneste



```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Tildeling

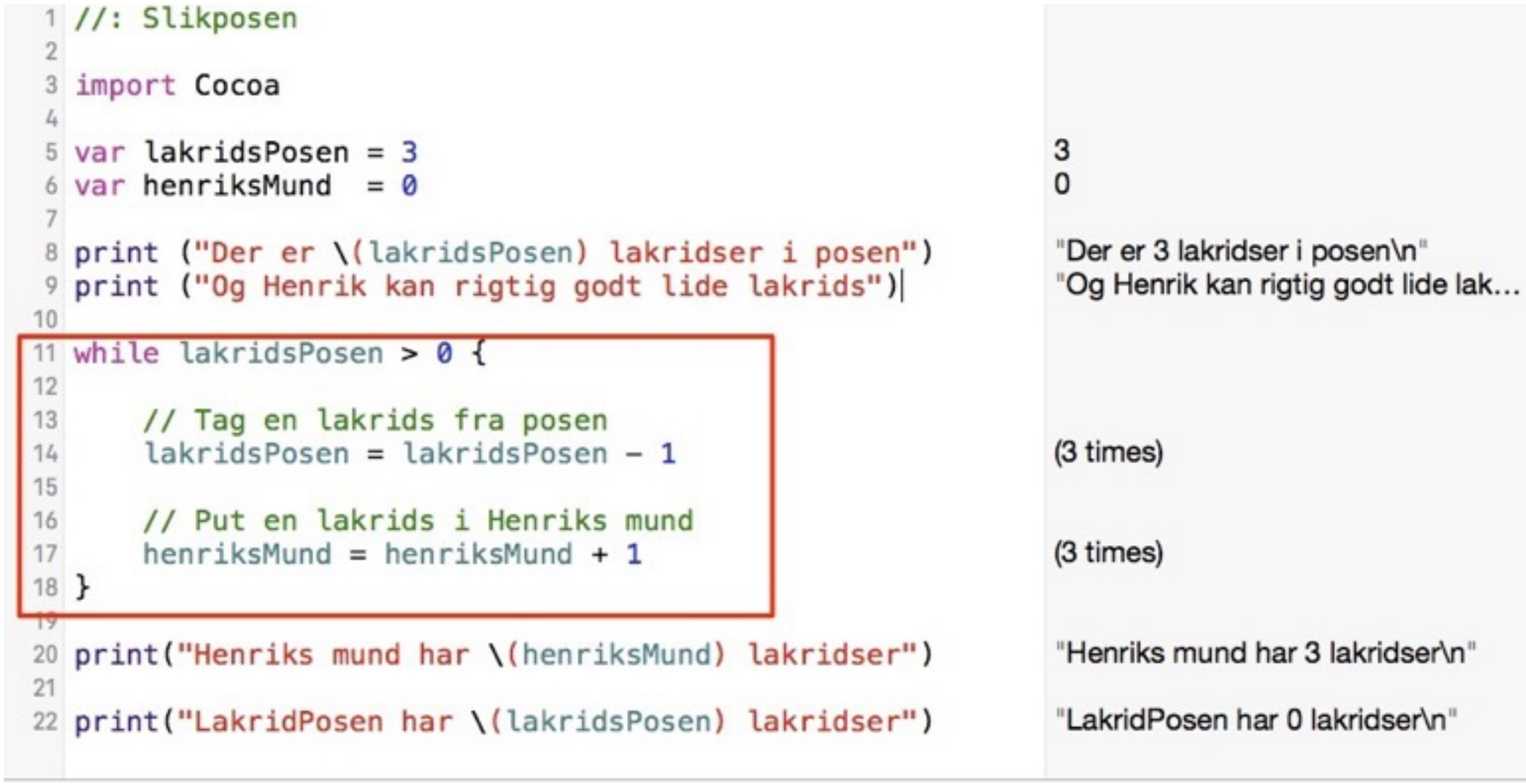
(Assignment)

```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Sammenligning

(Selektion)



```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


(3 times)


"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Gentagelse

(Iteration)

Tildeling - Computerens DNA

- En Bit er den mindste byggesten i en computer. Det er en kontakt.

- En kontakt / bit kan have 2 tilstande. Tændt eller slukket.

- Det er mængden af bits der afgør hvor mange informationer man kan gemme.

1 Bit - Sort / Hvid

- Har man kun 1 bit / kontakt så har man 2 muligheder.
- Tændt = Hvidt
- Slukket = Sort.

Rød - Gul - Grøn - Blå

- Vil man have kontakter til at vise grundfarverne skal man have 2 kontakter.

- Slukket / Slukket = Rød

- Slukket / Tændt = Gul

- Tændt / Slukket = Grøn

- Tændt / Tændt = Blå

Det binære tal system

- Hver gang man har en bit mere så fordobler man antallet af kombinationer man kan tildele værdier.

- Dette kaldes også for det binære talsystem

- Hver bit kan have værdien 0 / 1 (0 = slukket / 1 = tændt)

Bits og bytes

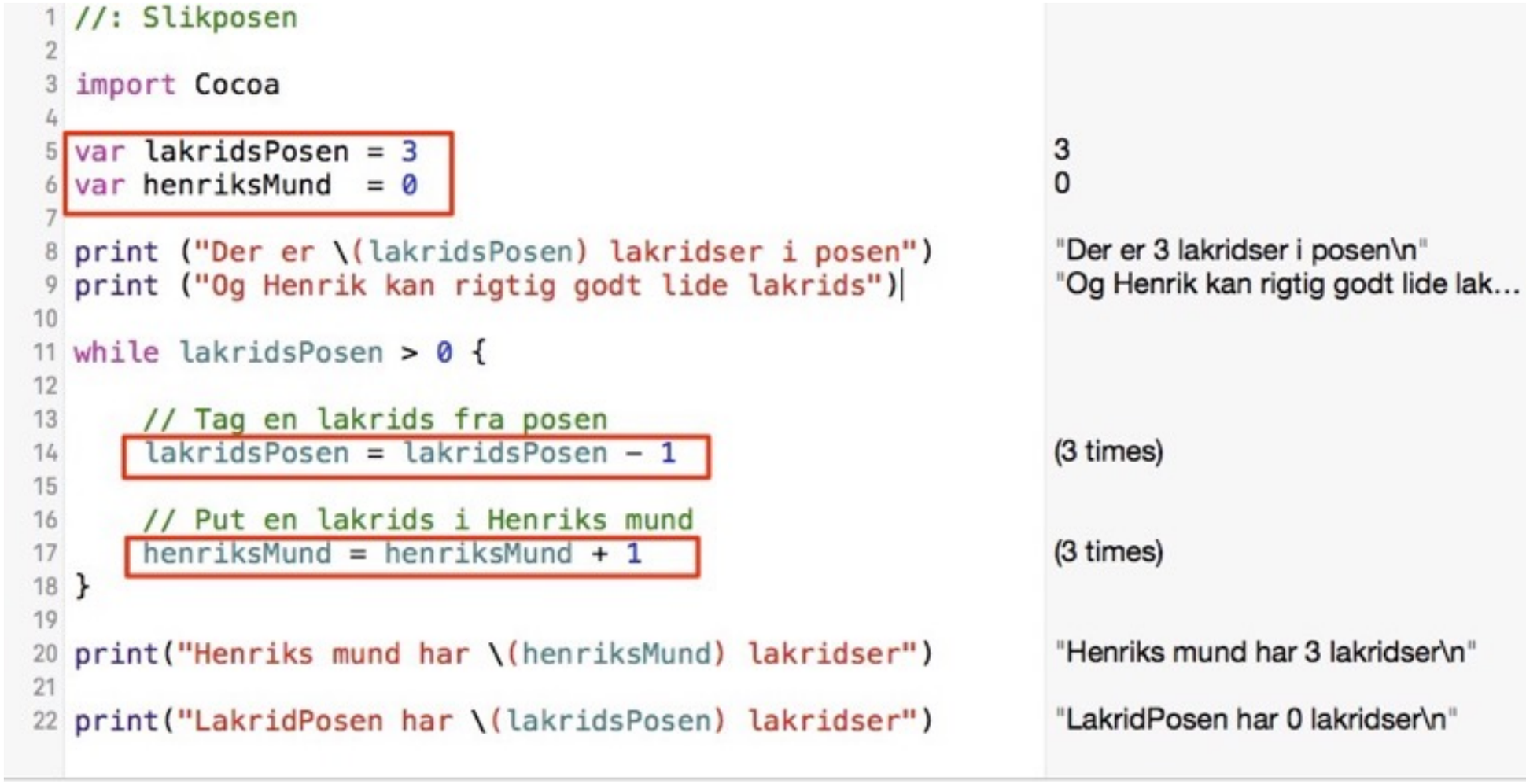
- Har man 8 bit har man en byte
- En byte = 256 kombinations muligheder af kontakter
- 0 0 0 0 0 0 0 0
- 128 - 64 - 32 - 16 - 8 - 4 - 2 - 1
- Hvis man har tallet 130 = 01000001 (binært)
- Hvad svarer 01110010 til ? (host host… 114)

32 bit / 64 bit

- Hvis en processor er 32 bit kan den håndtere instruktioner på en gang af 32 bits længde.

- 32 bit = 00000000 00000000 0000000 0000000 = 4294967296 kombinationer

- 64 bit = 00000000 00000000 0000000 0000000 00000000 00000000 00000000 0000000 = 1,84467440737096E19 kombinationer



```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Tildeling - Variabler

(Assignment)

Variabler

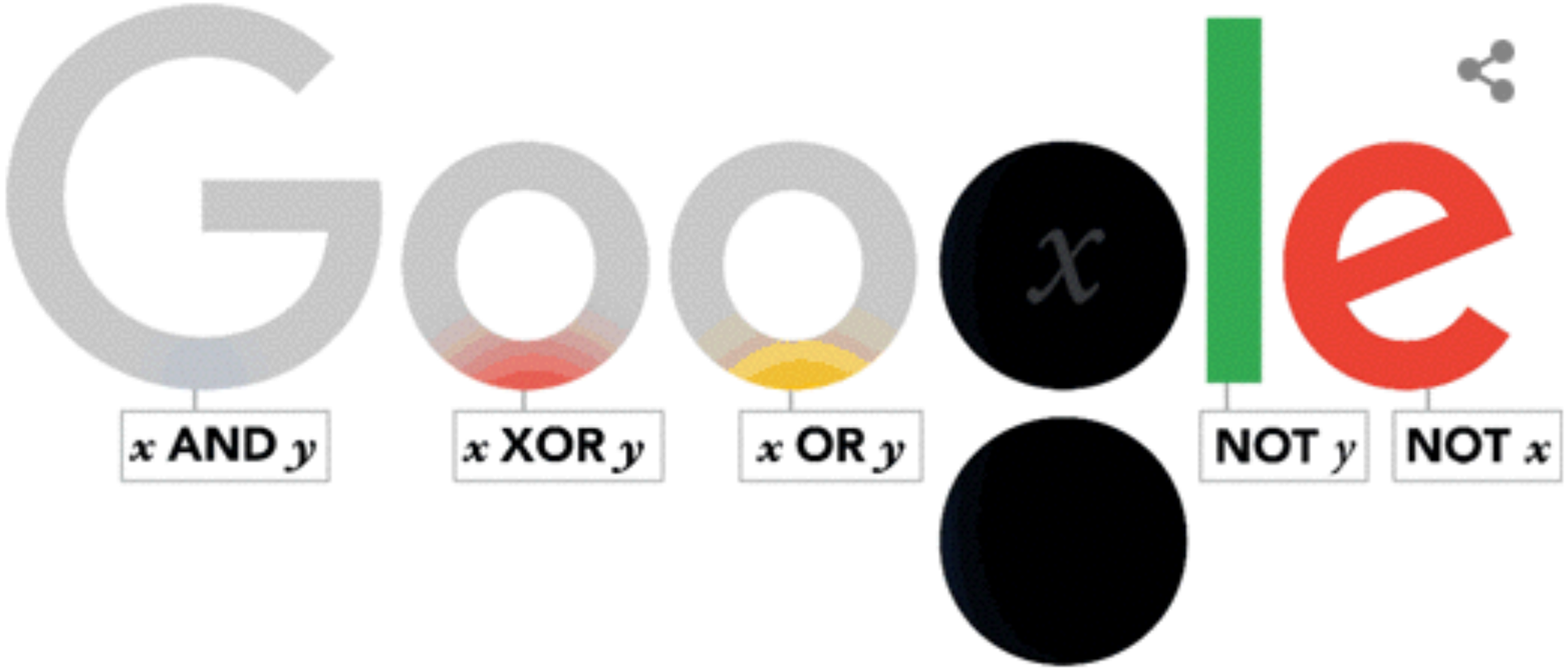
- var <variabelnavn>:<VariabelType> = <værdi>
- Eksempler:
- var navn:string
- var tal:int
- var kommatal:float

Swift kræver en intialisering

- Variabler skal have en værdi når de oprettes
- var tal:int
  tal = 4
- Vi kan undlade typen ved at skrive værdien direkte efter variabel navnet
- var tal = 4

Basistyper i Swift

- UInt = Unsigned Integer = heltal uden fortegn
- Int = Integer = heltal
- Float = Flydende punkt = kommatal (32 bit, 6 decimaler)
- Double = Dobbelt float = stort kommatal (64 bit, 15 decimaler)
- String = En streng = En tekst
- Bool = en kontakt = ja eller nej = true eller false



Google-søgning Jeg prøver lykken

Google.dk på: Føroyskt

George Boole

Fra Wikipedia, den frie encyklopædi

George Boole (2. november 1815 – 8. december 1864), var en engelsk filosof og matematiker, mest kendt for sin opfindelse og udvikling af Boolsk algebra.

Boole, som er født i Lincoln, England og døde i Cork, Irland, havde ikke nogen formel uddannelse og var autodidakt. Interessen for matematik havde han fra sin far, som var amatørmatematiker, men allerede som otteårig havde han overskredet faderens matematiske viden.

Som yngre interesserede Boole sig hovedsagelig for sprog, som eksempelvis gav sig udtryk i hans oversættelser, af latinsk poesi, til engelsk, som 12-årig samt at han som 18-årig, talte og skrev tysk, fransk og italiensk flydende.

De følgende år interesserede han sig hovedsagelig for matematik, og studerede matematiske værker, lånt fra den lokale teknologiske skole, blandt andet Isaac Newton's *Principia and the works of 18th and 19th century* samt værker af de franske matematikere Pierre-Simon Laplace og Joseph-Louis Lagrange. Han kunne på forbavsende kort tid forstå, bevise og bruge de matematiske formuleringer, som foranstående matematikere havde udviklet.

24 år gammel publicerede Boole afhandlingen *Researches on the Theory of Analytical Transformations*, som blev bragt i Cambridge Mathematical Journal.

I 1849 fik Boole en lærestol som professor i matematik på Quens College, Cork, nu, anno 2011, University College Cork.

Variabler og RAM

- Hver type fylder en mængde bytes
- Jo mere vi kan smide i typen jo flere bytes har vi brug for.
- Vi får en Adresse i hukommelsen og længden af typen bestemmer så hvor meget vi får tildelt.

Håndtering af RAM

- Operativ systemet håndterer at vi får adgang til RAM

- Swift compileren styrer hukommelsen for os.

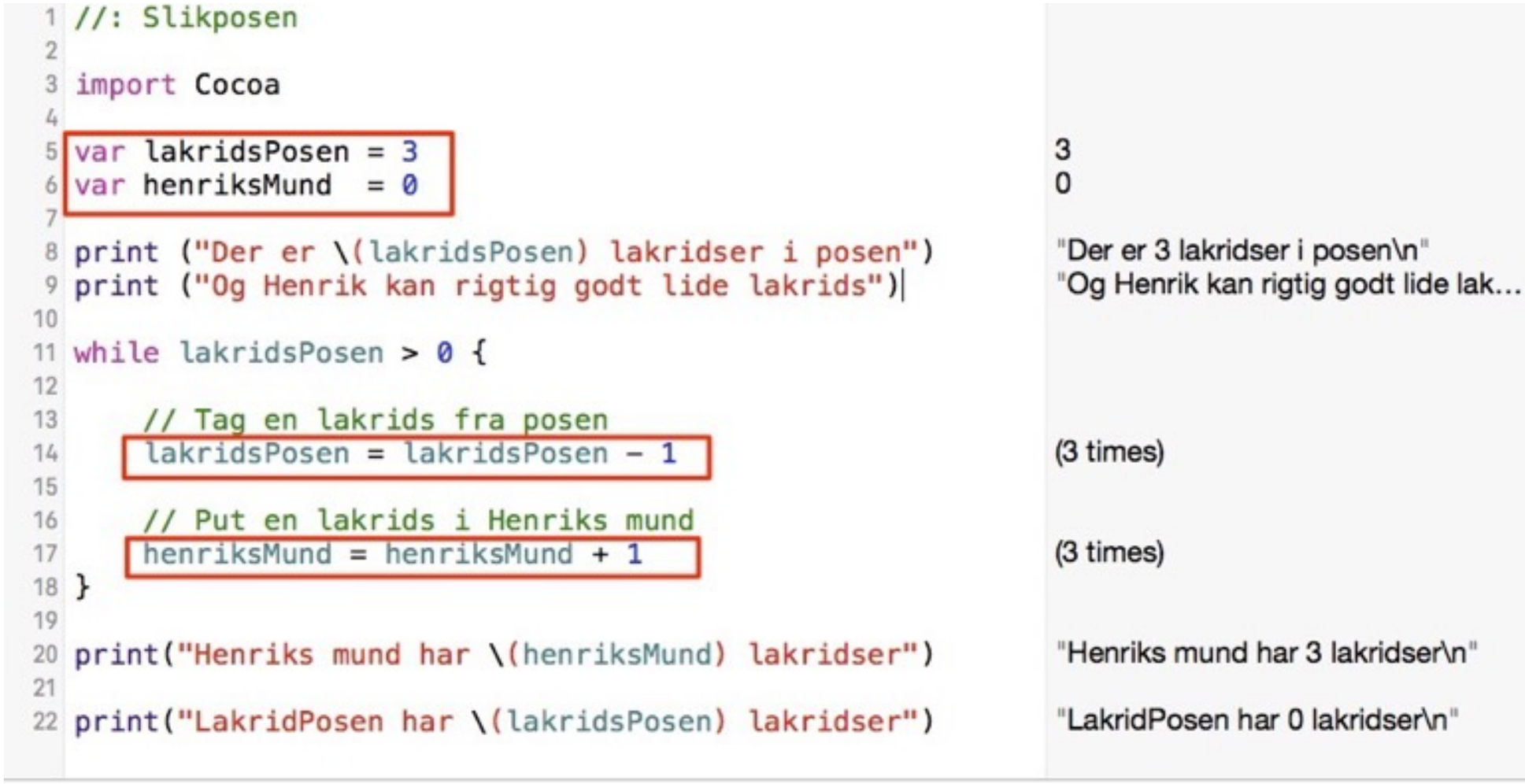
- Vi kan ikke lave en variabel i Swift og ikke bruge den. Så brokker compileren sig. (jeg kommer tilbage til hvad en compiler er)

Frigivelse af RAM

- Swift holder øje med når en variabel ikke bruges mere.

- ARC = Active Reference Counting

- Når der ikke er noget der bruger variablen mere så frigives den plads vi har reserveret fra hukommelsen.



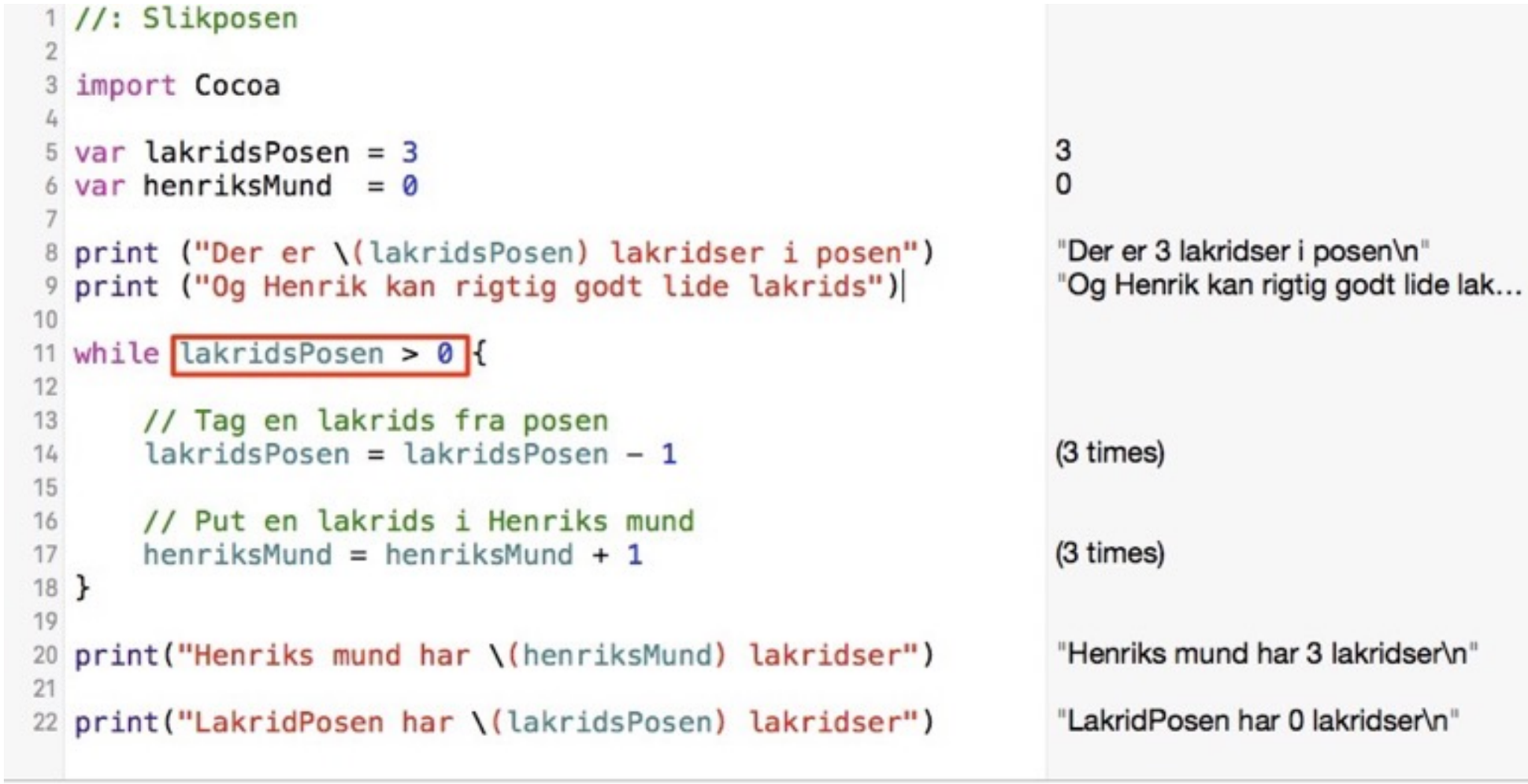
```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Tildeling - Variabler

Vi tager en bid ram.

Spørgsmål?



```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


(3 times)

"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Sammenligning

George Bool

Boolsk resultat

- Alle sammenligninger resulteret i et boolsk resultat.
- Sandt eller Falsk. true eller false.
- Gentage så længe antallet af lakridser i slikposen er mere end 0.
- Så længe der er slik i posen er resultatet af sammenligningen sandt = true = bit værdi 1.

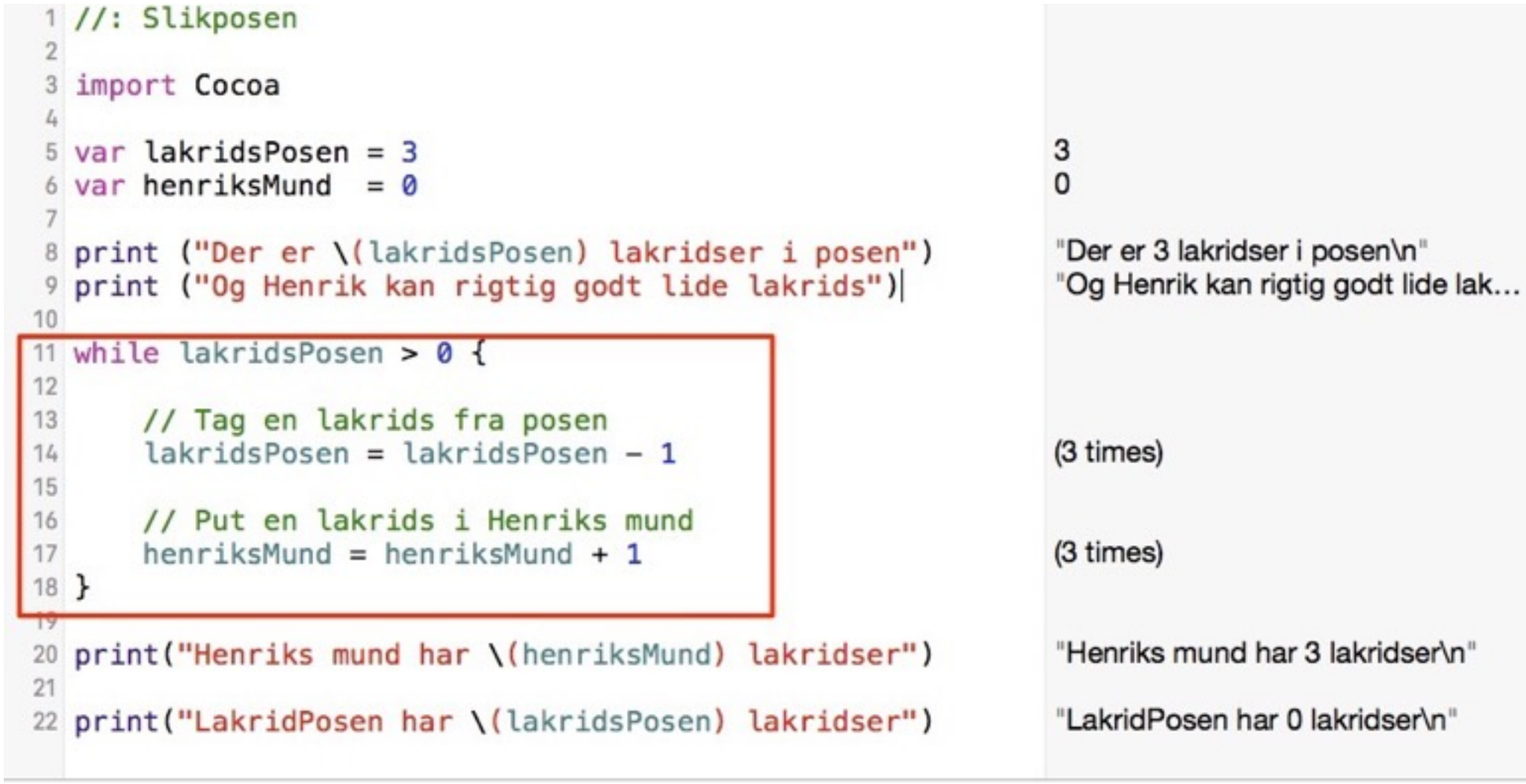
GATES

- AND gate (&& eller &)
- OR gate (|| eller |)
- NOT gate (!)
- XOR gate (Kun bitvis ^)

Eksempler

- slikPosen > 0
- (slikPosen > 0) && (henriksMund < 100)
- (slikPosen != 0) || (henriksMund < 100)

Spørgsmål til sammenligninger og boolske værdier?



```
1  //: Slikposen
2
3  import Cocoa
4
5  var lakridsPosen = 3
6  var henriksMund  = 0
7
8  print ("Der er \(lakridsPosen) lakridser i posen")
9  print ("Og Henrik kan rigtig godt lide lakrids")
10
11 while lakridsPosen > 0 {
12
13     // Tag en lakrids fra posen
14     lakridsPosen = lakridsPosen - 1
15
16     // Put en lakrids i Henriks mund
17     henriksMund = henriksMund + 1
18 }
19
20 print("Henriks mund har \(henriksMund) lakridser")
21
22 print("LakridPosen har \(lakridsPosen) lakridser")
```

```
3
0

"Der er 3 lakridser i posen\n"
"Og Henrik kan rigtig godt lide lak...



(3 times)


(3 times)


"Henriks mund har 3 lakridser\n"

"LakridPosen har 0 lakridser\n"
```

Gentagelse

(Iteration)

Sådan fungerer et udviklingsmiljø

4 Generation / 5 Generation

3 Generation

Assembler

CPU
kode
binær

Kompileren

- 1. Compileren “parser” og forstår vores program kodes syntax, kontrollere for tekniske fejl.

- 2. Compileren linker alle biblioteker (**import**) ind som skal bruges

- 3. Compileren laver maskinkode til afvikling på hardwaren

- 4. (Fra iOS9 laves der ikke ren maskinkode mere - men det skal vi ikke tænke så meget over nu)

Swift er type sikker

- Swift kompileren melder fejl når typer ikke er ens.

- let etTal = 10
  let etKommatal = 20.0

  let etResultat = etTal + etKommatal <— Fejl

- let etResultat = etKommatal + Double(etTal)

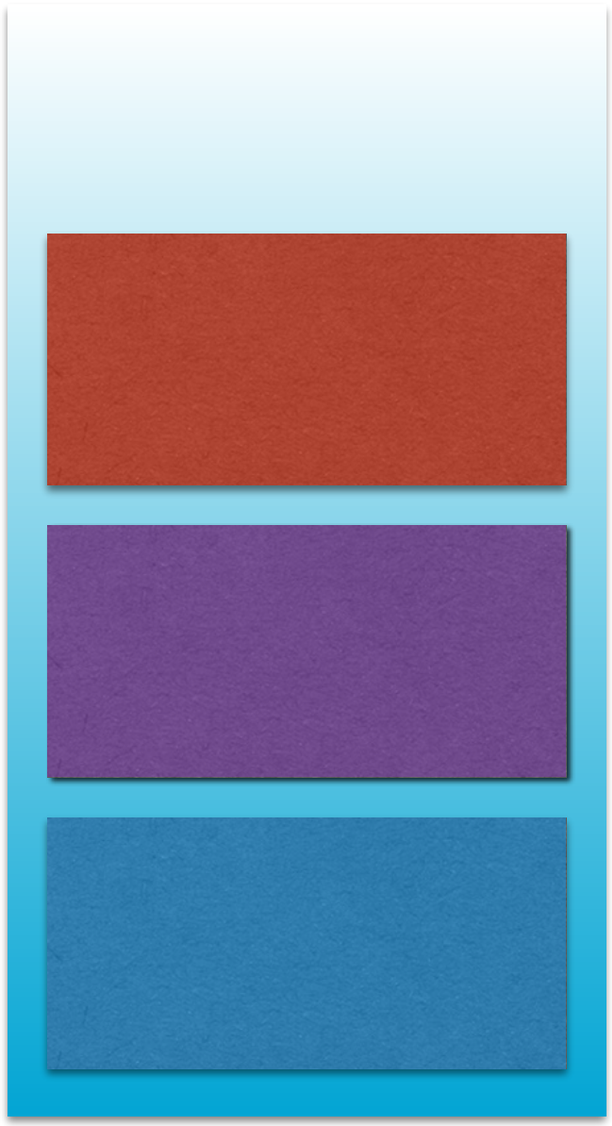
Typecasting

- <Type>(<variabel>)

Eksempel i Xcode Playgrounds.

Stak (stack)

Push



Pop

Array

- Et array er en struktur der kan indeholde flere variabler af samme type.

- var navne = [“Henrik”, “Mads”, “Claudia”]

- var navne:[string]

Eksempel i xcode playgrounds

Welcome to Xcode

Version 7.0.1 (7A1001)

Øvelser

aftenskole@bidblog.dk

Brug denne email til spørgsmål om lektioner!
Brug udvikler sparring til at sparre med dine med kursister