Programmering og Problemløsning

4.1: Kaldestakken, bunken, referenceceller, højere-ordens og anonyme funktioner

Repetition af Nøglekoncepter

- Virkefelter
- Funktioner
- Programmer ‘baglæns’
- Dokumentation
- Løkker

- Tupler
- Betingelser

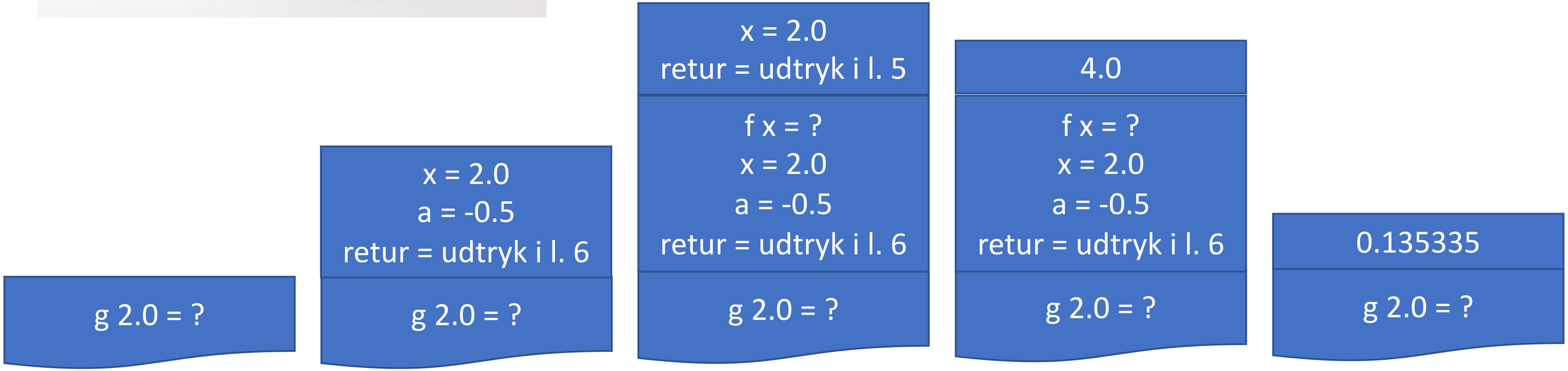
```
let fib N =
  let mutable pair = (1,1)
  for i = 3 to N do
    pair <- (snd pair, fst pair + snd pair)
  snd pair

let N = 5
printfn "%d: %d" N (fib N)
```

Kald-stakken (værdier og variable)

Stakken (The Stack)

```
1 let f x =
2   x*x
3 let g x =
4   let a = -1.0/2.0
5   exp (a * f x)
6 printfn "%g" (g 2.0)
```



g 2.0 = ?

x = 2.0
a = -0.5
retur = udtryk i l. 6

g 2.0 = ?

x = 2.0
retur = udtryk i l. 5

f x = ?
x = 2.0
a = -0.5
retur = udtryk i l. 6

g 2.0 = ?

4.0

f x = ?
x = 2.0
a = -0.5
retur = udtryk i l. 6

g 2.0 = ?

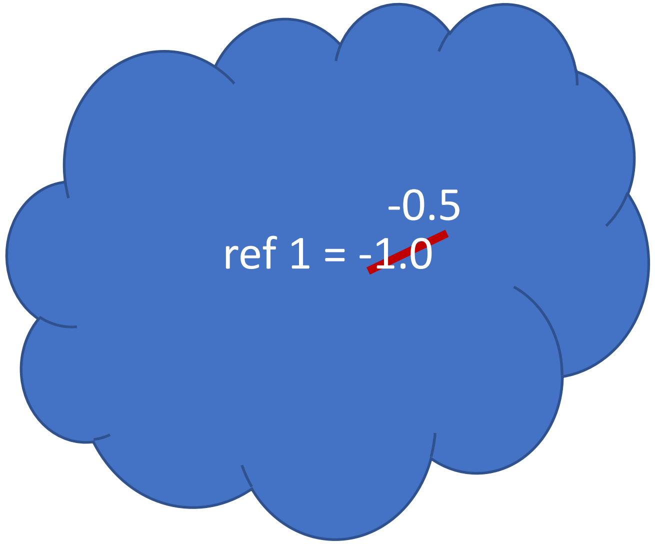
0.135335

g 2.0 = ?

Referenceceller

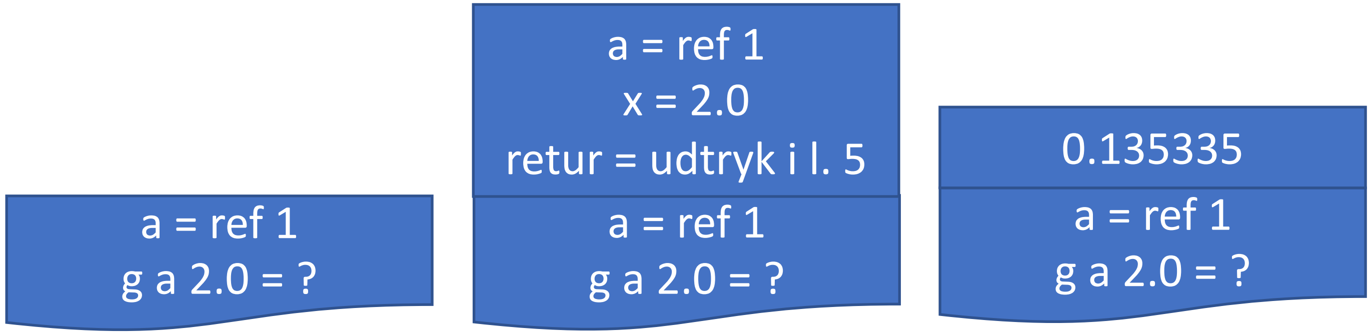
Bunken (The Heap)

```
1 let g a x =
2   a := -1.0/2.0
3   exp (!a * x * x)
4 let a = ref -1.0
5 printfn "%g" (g a 2.0)
6 printfn "%g" !a
```



-0.5

ref 1 = ~~-1.0~~



a = ref 1
g a 2.0 = ?

a = ref 1
x = 2.0
retur = udtryk i l. 5

a = ref 1
g a 2.0 = ?

0.135335

a = ref 1
g a 2.0 = ?

Aliasing (undgå!)

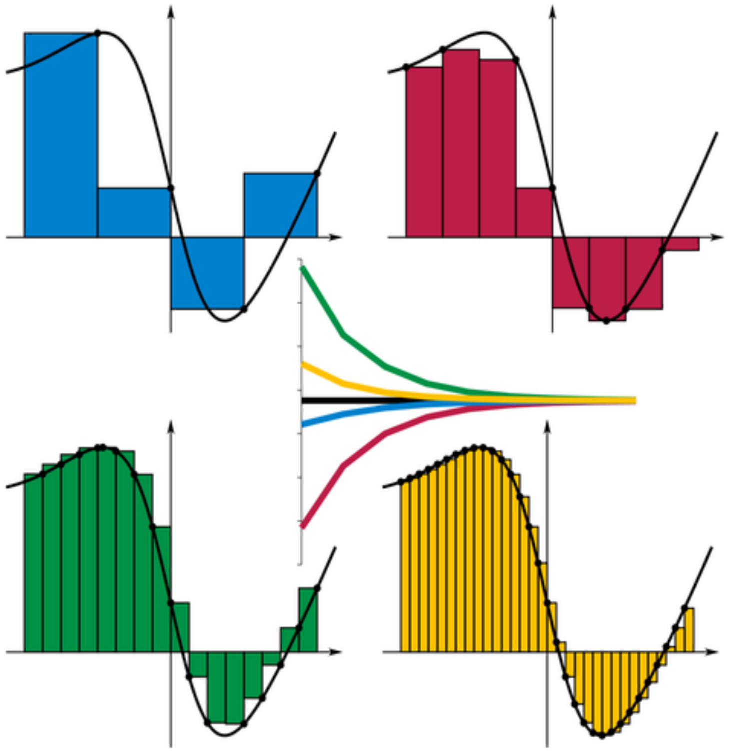
```
> let a = ref 1.0
- let b = ref 2.0
- let c = a
- printfn "a = %g, b = %g, c = %g" !a !b !c
- b := 3.0
- c := 4.0
- printfn "a = %g, b = %g, c = %g" !a !b !c;;
a = 1, b = 2, c = 1
a = 4, b = 3, c = 4
val a : float ref = {contents = 4.0;}
val b : float ref = {contents = 3.0;}
val c : float ref = {contents = 4.0;}
val it : unit = ()
```

Værdien c er samme reference til bunken som a

Ændrer hvad reference peger på

Indholdet af a ændrede sig indirekte!

Højere-ordens funktioner

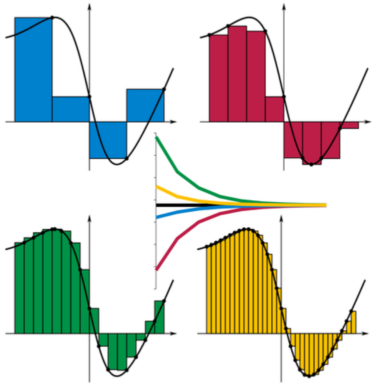


By I, KSmrq, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=2347919

```
/// Estimate the integral of f
/// from a to b with stepsize d
let integrate f a b d =
  let mutable sum = 0.0
  let mutable x = a
  while x < b do
    sum <- sum + d * (f x)
    x <- x + d
  sum

let a = 0.0
let b = 1.0
let d = 0.01
let result = integrate exp a b d
printfn "Int_%g^%g exp(x) dx = %g" a b result
```

Højere-ordens funktioner



By I, KSmrq, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=2347919

```
/// Estimate the integral of f
/// from a to b with stepsize d
let integrate f a b d =
  let mutable sum = 0.0
  let mutable x = a
  while x < b do
    sum <- sum + d * (f x)
    x <- x + d
  sum

let a = 0.0
let b = 1.0
let d = 0.01
let result = integrate exp a b d
printfn "Int_%g^%g exp(x) dx = %g" a b result
```

```
/// Estimate the integral of f
/// from a to b with stepsize d
let integrate f a b d =
  let mutable sum = 0.0
  let mutable x = a
  while x < b do
    sum <- sum + d * (f x)
    x <- x + d
  sum

let a = 0.0
let b = 1.0
let truth = exp 1.0 - 1.0
for e = 0 to 6 do
  let d = 10.0**(float -e)
  let result = truth - integrate exp a b d
  printfn "d = %e: exp 1.0 - 1.0 - Int_%g^%g exp(x) dx = %g" d a b result
```

Anonyme funktioner

```
let f x = x * exp(x)
f 3.0
```

```
let f = fun x -> x * exp(x)
f 3.0
```

```
/// Estimate the integral of f
/// from a to b with stepsize d
let integrate f a b d =
  let mutable sum = 0.0
  let mutable x = a
  while x < b do
    sum <- sum + d * (f x)
    x <- x + d
  sum

let a = 0.0
let b = 1.0
let d = 1e-5
let result = integrate (fun x -> x * exp(x)) a b d
printfn "Int_%g^%g f(x) dx = %g" a b result
```

DIKU Bits

MONDAY LECTURES
BLOCK 1, 2018

Tid: 24. september 2018 kl. 12.15-13.00
Sted: Lille UP1

24 SEPTEMBER

Compositionality in reversible programming

Robin Kaarsgaard
Postdoc in the PLTC section