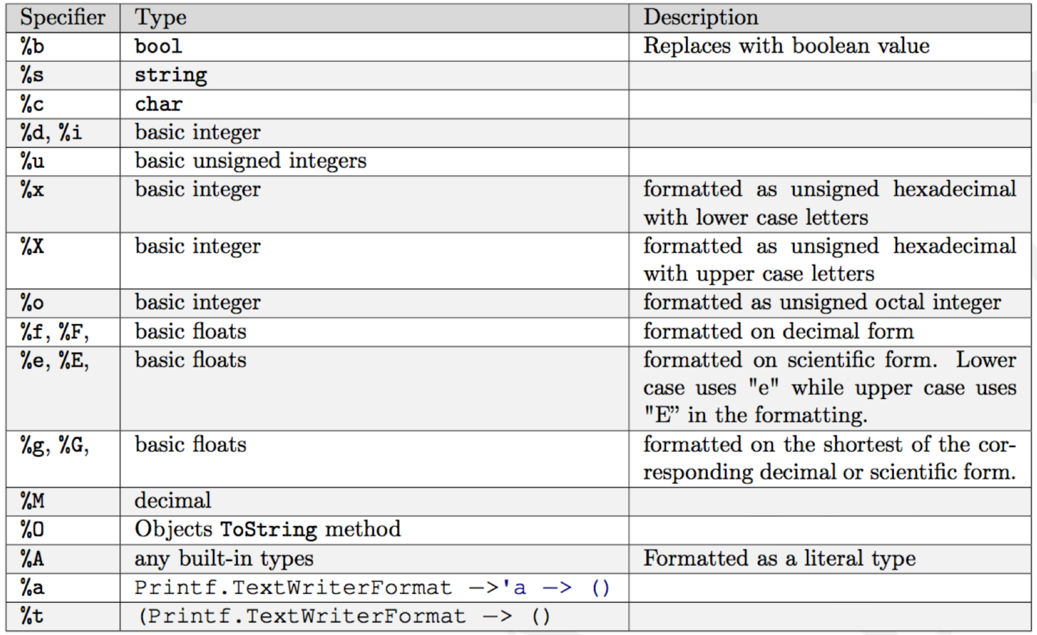
Programmering og Problemløsning

3.1: Funktioner, dokumentation og løkker

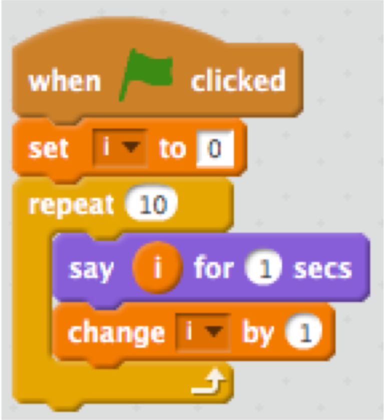
Repetition af Nøglekoncepter

- Præcedens og association
- Verbose og letvægtssyntaks
- Virkefelter
- Nøgleord

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



Specifier	Type	Description
%b	bool	Replaces with boolean value
%s	string	
%c	char	
%d, %i	basic integer	
%u	basic unsigned integers	
%x	basic integer	formatted as unsigned hexadecimal with lower case letters
%X	basic integer	formatted as unsigned hexadecimal with upper case letters
%o	basic integer	formatted as unsigned octal integer
%f, %F,	basic floats	formatted on decimal form
%e, %E,	basic floats	formatted on scientific form. Lower case uses "e" while upper case uses "E" in the formatting.
%g, %G,	basic floats	formatted on the shortest of the corresponding decimal or scientific form.
%M	decimal	
%O	Objects ToString method	
%A	any built-in types	Formatted as a literal type
%a	Printf.TextWriterFormat ->'a -> ()	
%t	(Printf.TextWriterFormat -> ()	



```
when clicked
set i to 0
repeat 10
    say i for 1 secs
    change i by 1
```

https://tinyurl.com/y923467c

https://tinyurl.com/y8yuuyy4

Fibonacci

For-løkke

```
let mutable m = 1
let mutable n = 1
let N = 5
for i = 3 to N do
  let p = m + n
  m <- n
  n <- p
printfn "%d: %d" N n
```

While-løkke

```
let mutable m = 1
let mutable n = 1
let mutable i = 3
let N = 5
while i <= 5 do
  let p = m + n
  m <- n
  n <- p
  i <- i + 1;
printfn "%d: %d" N n
```

Tupler

```
$fsharpi
...
> let a = (1, 1.0);;
val a : int * float = (1, 1.0)
```

Produktttype

Funktioner til at indicerer i par

```
> printfn "%A %A" (fst a) (snd a);;
1 1.0
val it : unit = ()
```

Parentes unødvendig men anbefalelses

```
> let b = 1, "en", '\049'
val b : int * string * char = (1, "en", '1’)
```

Venstre side af en binding kan have navngivne tuple-elementer

```
> let (b1, b2, b3) = b;;
val b3 : char = '1'
val b2 : string = "en"
val b1 : int = 1
```

Hele typen - ikke enkelt - elementer kan være mutérbare

```
> let mutable c = (1,2)
- c <- (2,3)
- printfn "%A" c;;
(2, 3)
val mutable c : int * int = (2, 3)
val it : unit = ()
```

Fibonacci

For-løkke

```
let mutable m = 1
let mutable n = 1
let N = 5
for i = 3 to N do
  let p = m + n
  m <- n
  n <- p
printfn "%d: %d" N n
```

While-løkke

```
let mutable m = 1
let mutable n = 1
let mutable i = 3
let N = 5
while i <= 5 do
  let p = m + n
  m <- n
  n <- p
  i <- i + 1;
printfn "%d: %d" N n
```

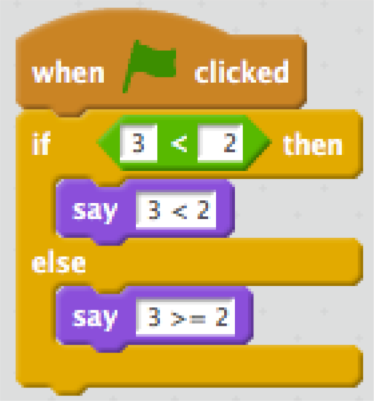
Tupple + for-løkke

```
let mutable pair = (1,1)
let N = 5
for i = 3 to N do
  pair <- (snd pair, fst pair + snd pair)
printfn "%d: %d" N (snd pair)
```

```
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)
```

Betingelser



```
when clicked
if 3 < 2 then
  say 3 < 2
else
  say 3 >= 2
```

If-then-else

```
if 3 < 2 then
  printfn "3 < 2"
else
  printfn "3 >= 2";;
3 >= 2
val it : unit = ()

let str =
  if 3 < 2 then
    "3 < 2"
  else
    "3 >= 2";;
val str : string = "3 >= 2"
```

Kæde af betingelser

```
let str =
  if 3 < 2 then
    "3 < 2"
  elif 3 = 2
    "3 = 2"
  else
    "3 > 2";;
val str : string = "3 > 2"
```

Decimal til Binær

```
let N = 116
let mutable n = N
let mutable str = ""
while n > 0 do
  let rest = n % 2
  n <- n / 2
  if rest > 0 then
    str <- "1"+str
  else
    str <- "0"+str
printfn "%d_10 = %s_2" N str
```

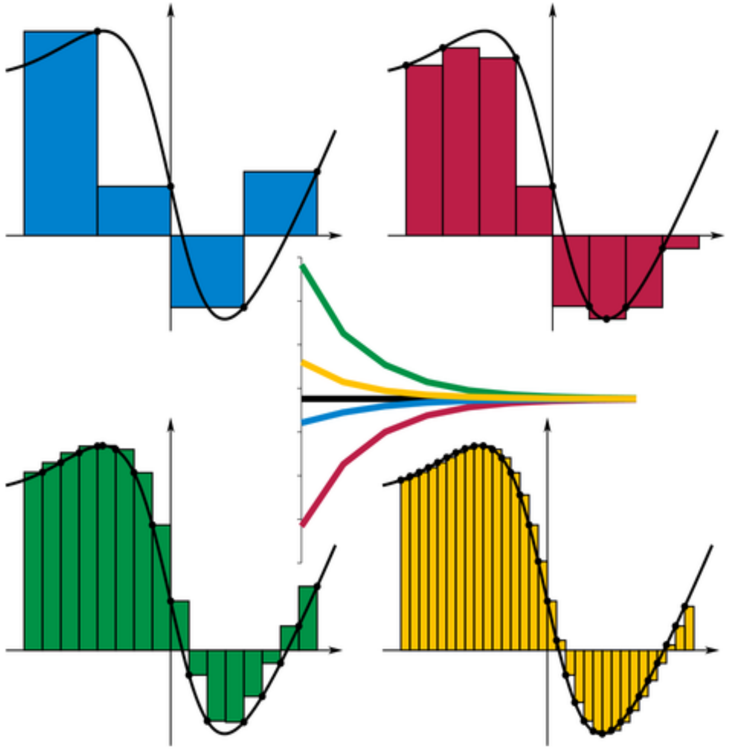
```
let N = 116
let mutable n = N
let mutable str = ""
while n > 0 do
 str <- (if n % 2 > 0 then "1" else "0") + str
 n <- n / 2
printfn "%d_10 = %s_2" N str
```

Hvad gør programmet?

i på højre side er altid 0

```
let i = 0
while i < 3 do
  let i = i + 1
  printfn "%d" i
```

Højere ordens funktioner

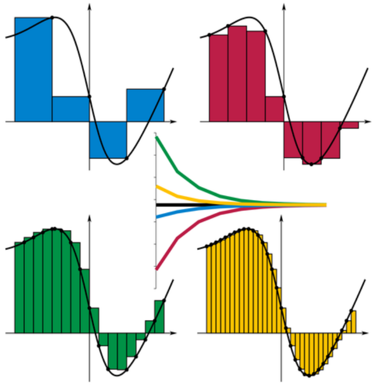


By I, KSmrq, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=2347919

```
/// 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