

Programmering og Problemløsning

3.3: Tupler, betingelser, højere-ordens funktioner

Repetition af Nøglekoncepter

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

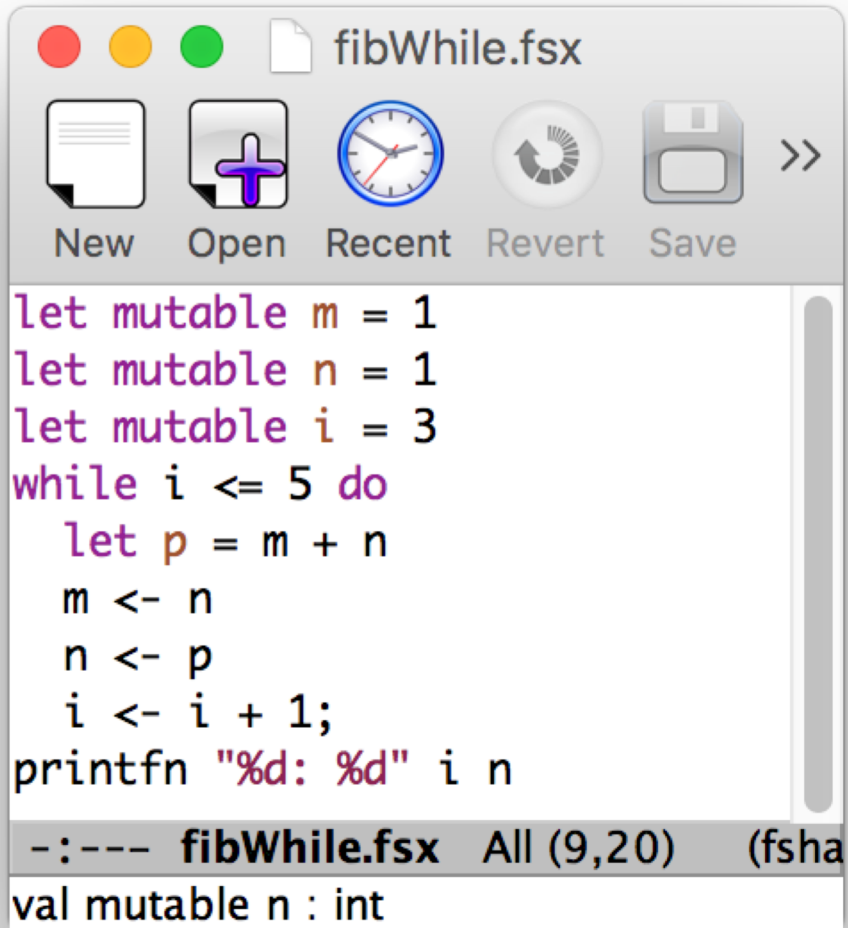
- Virkefelter
- Funktioner
- Programmer 'baglæns'
- Dokumentation
- Løkker

| 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 -> () | |



Repetition af Nøglekoncepter

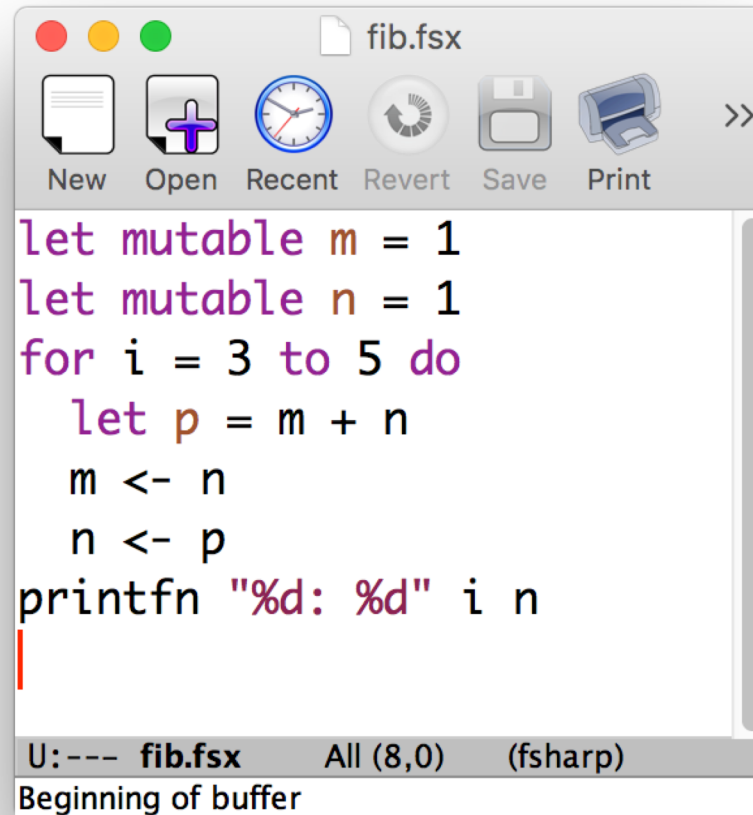
<https://tinyurl.com/y923467c>



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

-:--- fibWhile.fsx All (9,20) (fsharp)
val mutable n : int
```

<https://tinyurl.com/y8yuuyy4>



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

U:--- fib.fsx All (8,0) (fsharp)
Beginning of buffer
```

```
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
```

Tupler

```
$fsharpi
```

```
...
```

```
> let a = (1, 1.0);;
```

```
val a : int * float = (1, 1.0)
```

Produkttype

Funktioner til at
indicerer i par

```
> printfn "%A %A" (fst a) (snd a);;
```

```
1 1.0
```

```
val it : unit = ()
```

Parentes unødvendig
men anbefales

```
> 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
```

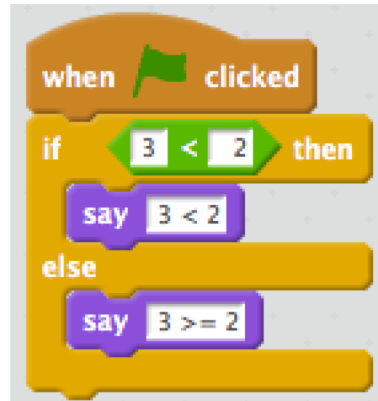
Tupple + for-løkke

```
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



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 then
-       "3 = 2"
-   else
-       "3 > 2";;
val str : string = "3 > 2"
```

Decimal til Binær

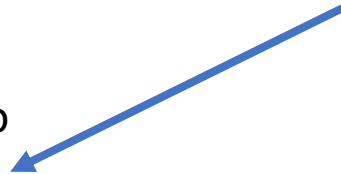
```
let N = 116
let mutable n = N
let mutable str = ""
while n > 0 do
    let rest = n % 2
    n <- n / 2
    if rest > 0 then
        str <- "1"+str
    else
        str <- "0"+str
printfn "%d_10 = %s_2" N str
```

```
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?

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

i på højre side er altid 0



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

