Introduktion til Programmering og Problemløsning (PoP)

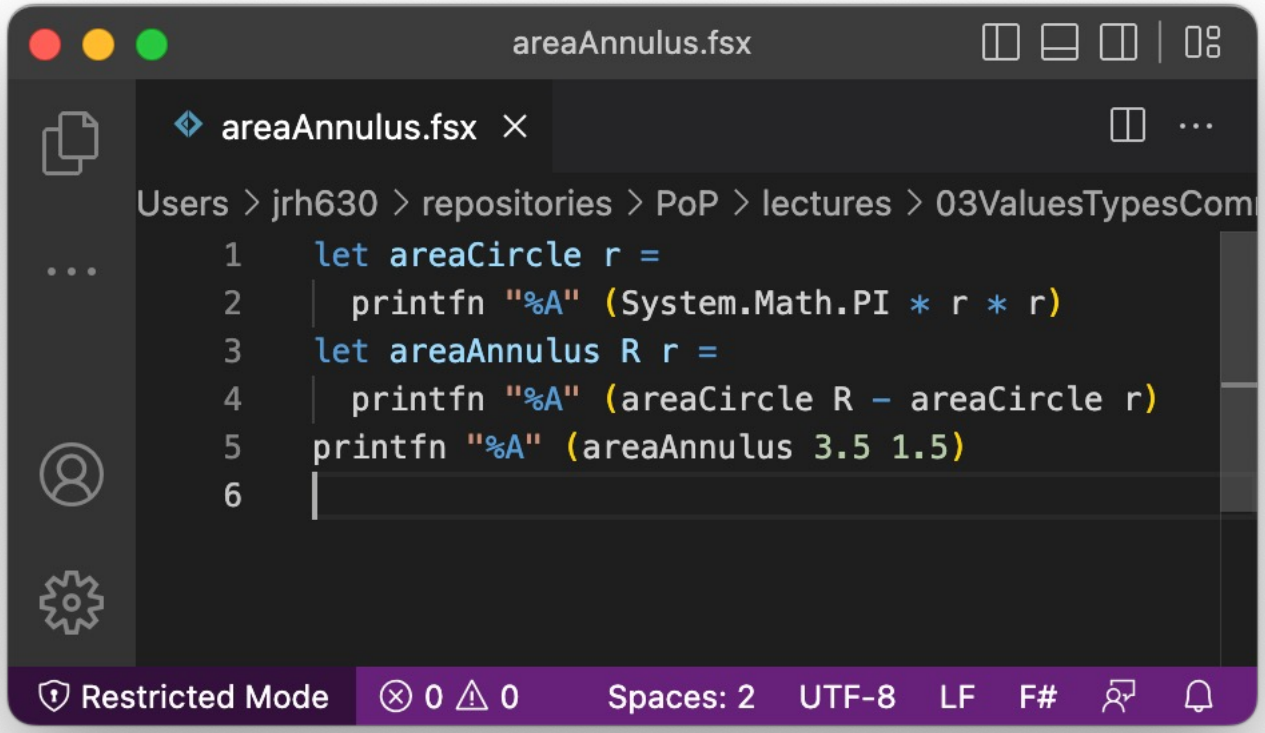
Option typer

Jon Sporring
Department of Computer Science
2022/09/12

UNIVERSITY OF COPENHAGEN

Besvar følgende med den du sidder ved siden af:

https://tinyurl.com/yykhuczh

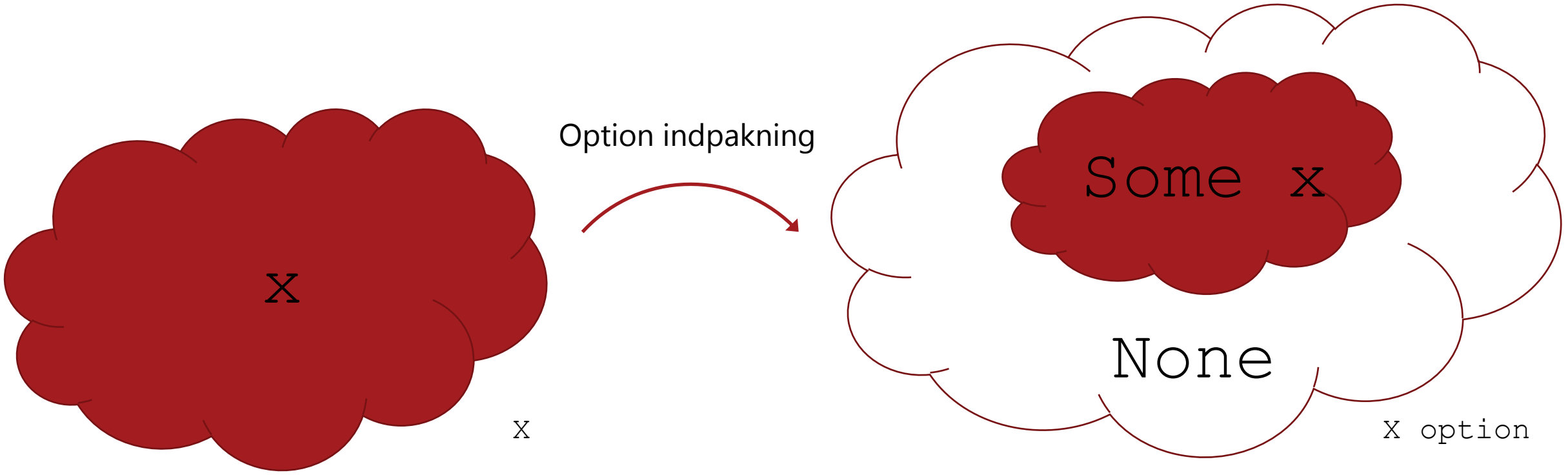


Servicemeddelelse: De næste kapitler i bogen er klar (med smårettelser til de nuværrende)

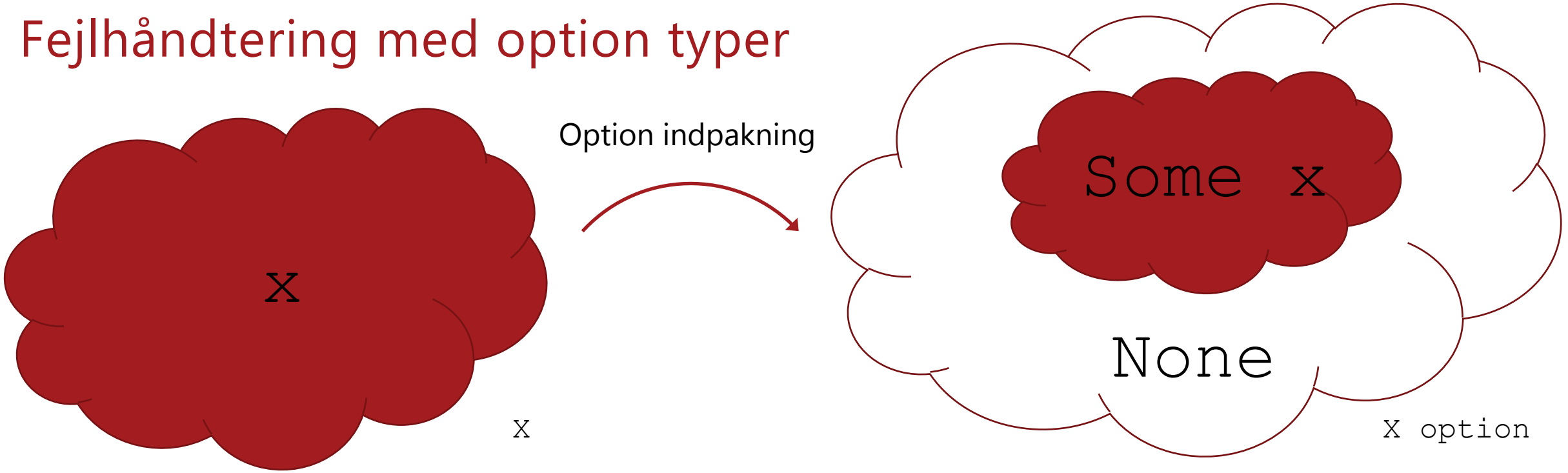
Fejlhåndtering med option typer

Divisions med 0

```
let div a b = a/b
div 3 0;;
System.DivideByZeroException: Attempted to divide by zero.
   at <StartupCode$FSI_0002>.$FSI_0002.main@() in
/Users/jrh630/repositories/PoP/lectures/03ValuesTypesComments/stdin:line 2
Stopped due to error
```



Fejlhåndtering med option typer

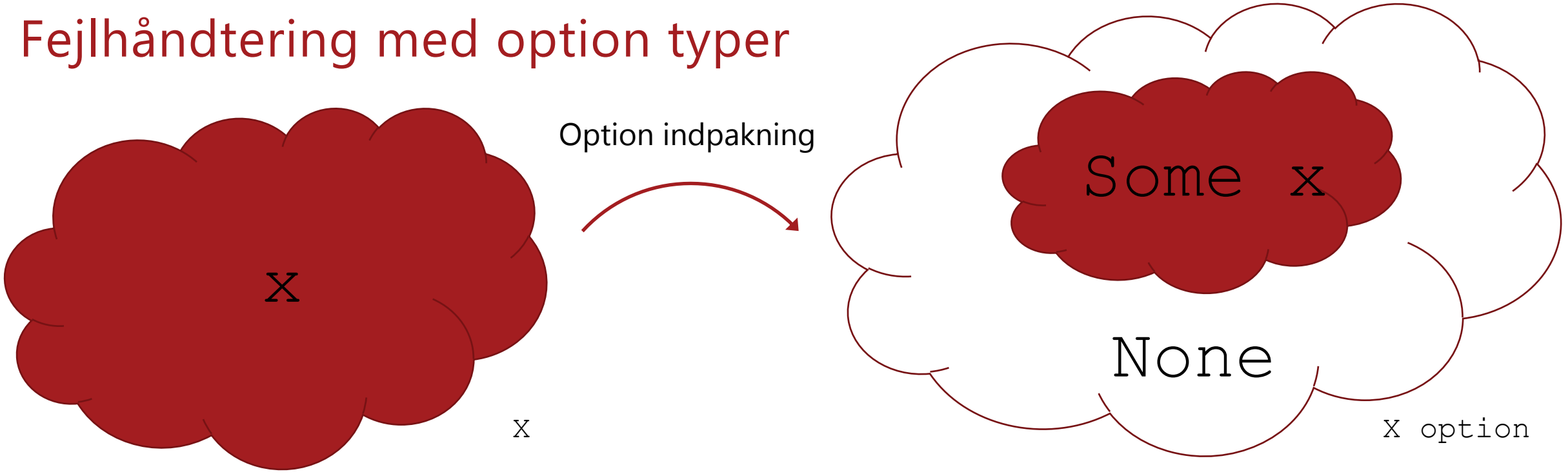



```
type packet = Error | Value of int;;
type packet =
  | Error
  | Value of int

let a = Error;;
val a: packet = Error
let b = Value 3;;
val b: packet = Value 3
```

Option typen kan holde vilkårlig type

Fejlhåndtering med option typer




```
let div a b =
  match b with
    0 -> None
    | _ -> Some (a/b)
div 3 0;;
val div: a: int -> b: int -> int option
val it: int option = None
```

```
let print (x: 'a option) =
  match x with
    None -> printfn "Divide by 0 error"
    | Some y -> printfn "%A" y
div 3 0 |> print
div 3 2 |> print;;
Divide by 0 error
1
val print: x: 'a option -> unit
val it: unit = ()
```

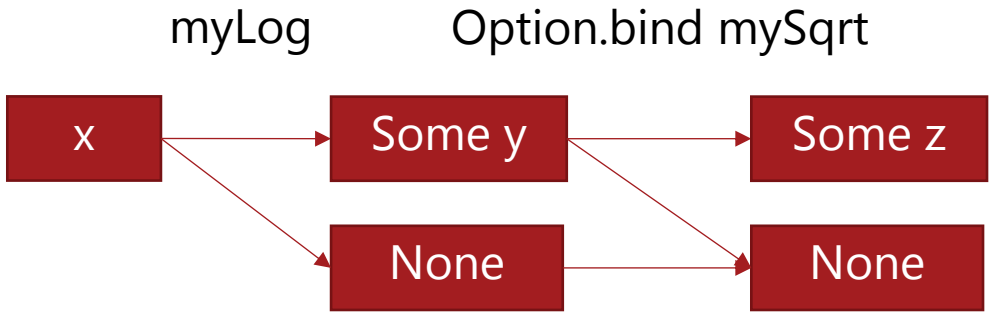
Sammensætning af funktioner f: 'a -> 'b option

Følgende funktioner håndterer evt. fejl med option-typer:

```
let myLog (x: float) : float option =
  if x > 0.0 then Some (log x) else None
let mySqrt (x: float) : float option =
  if x >= 0.0 then Some (sqrt x) else None;;
```

Direkte sammensætning er besværlig:

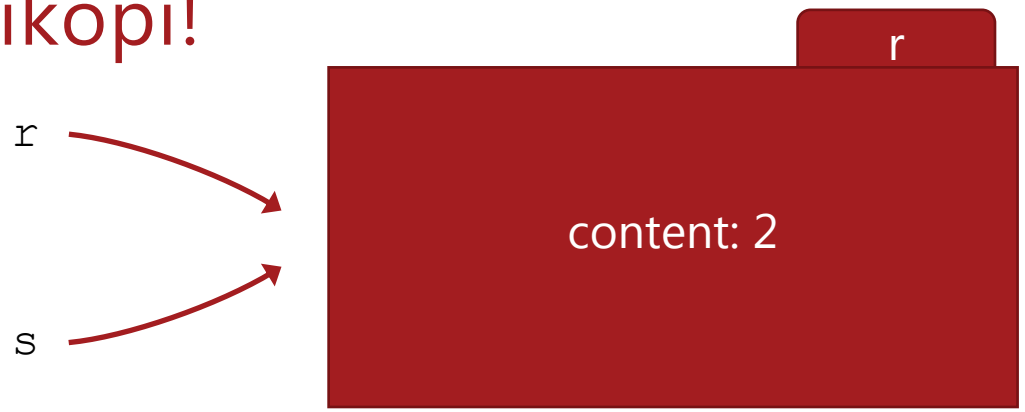
```
let mySqrtLog x : float option =
  let logX = myLog x
  match logX with
    None -> None
    | Some y -> mySqrt y
mySqrtLog 1.0;;
val it : float option = Some 0.0
```



Denne funktionalitet er allerede tilgængelig med Option.bind funktionen:

```
val bind : (('a -> 'b option) -> 'a option -> 'b option)
1.0 |> myLog |> Option.bind mySqrt
val it : float option = Some 0.0
```

Aliasing: referencekopi og ikke værdikopi!



Records er en referencetype

```
type record = {mutable content: int}
let r = {content = 1};;
```

```
type record =
  { mutable content: int }
val r: record = { content = 1 }
let s = r
r.content <- 2
printfn "%A" s;;
{ content = 2 }
val s: record = { content = 2 }
val it: unit = ()
```

Fordel:
- Referencekopiering er hurtigt

Bagdel:
- Aliasing = sideeffekt

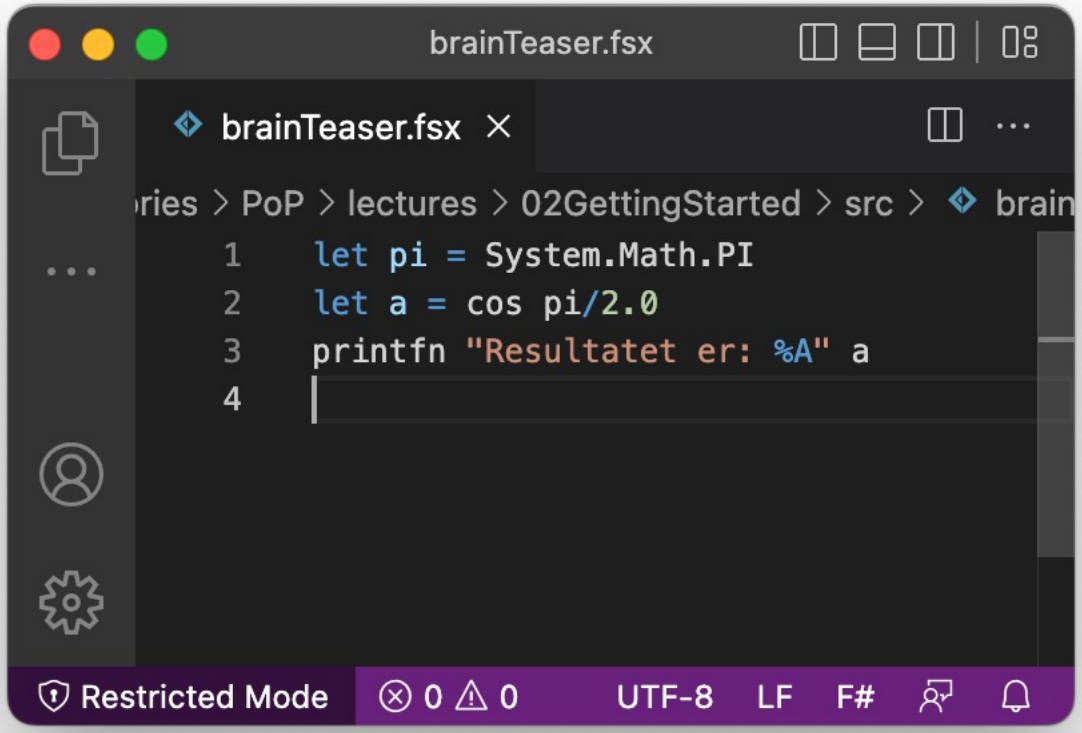
Kur
- Dyb kopi

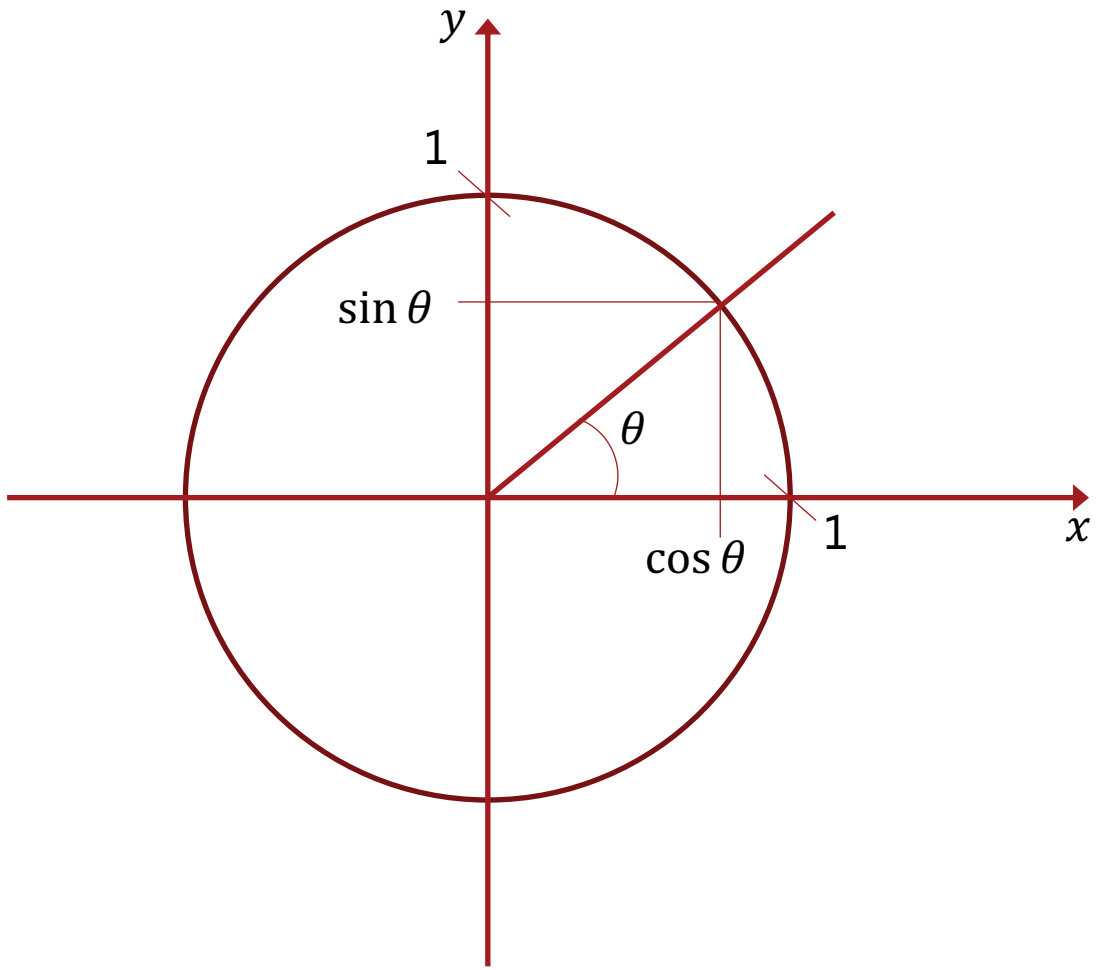
```
let copy (r: record) : record =
  {content = r.content}

let s = copy r
r.content <- 2
printfn "%A %A" r s;;
{ content = 2 } { content = 1 }
```

Mandagens hjernevrider

https://tinyurl.com/242jdzmw





Resumé

I dag har vi talt om:

- Fejlhåndtering med option typer
- Records og aliasing
- Mandagsopgaven