Kunnskap for en bedre verden

TDT4105 Informasjonsteknologi grunnkurs:
Uke 43:

Datastrukturer (kap. 8)

Anders Christensen
anders@ntnu.no

Rune Sætre
satre@ntnu.no

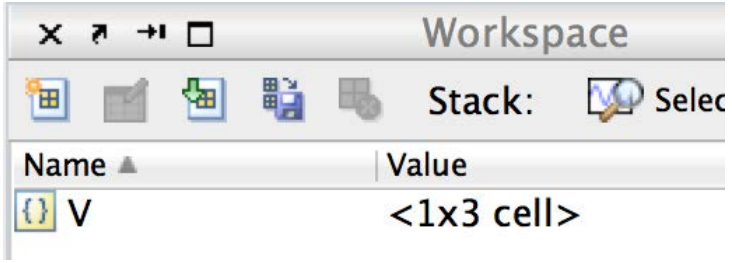
Læringsmål og pensum

- Læringsmål
  - Datastrukturer:
    - Cell Arrays
    - Structures
  - Problemløsing og litt repetisjon
  - return og break – overstyring av normal programflyt
- Pensum
  - Kapittel 8 Data Structures

Datastrukturer

- Datatype:
  - Datatypen til en variabel er de verdiene som variabelen kan ha
  - Har ofte et sett operasjoner som kan utføres på variabler av den aktuelle datatypen.
  - single, double, int8, int16, int32, int64, char, string, logical
- Datastruktur:
  - Samling variabler, gjerne av ulik datatype, forbundet på ulike måter
  - For eksempel: Vektorer og matriser
    - Alle elementer av samme type
    - Fast struktur (alle rader har like mange kolonner)

Cell Arrays



- Vektorer/Matriser der elementene er cells
  - CellArray kan lagre data av ulike typer
- Indekseres som ordinære vektorer/matriser
- Bruker *sløyfe/krøll-parenteser:* {}
  - V = {1910, 'NTNU', 'Trondheim'}
  - Innholds-indeksering:
  - V{1} = 1910 (tall, double)
  - V{2} = 'NTNU' (string, 4 tegn lang)
  - V{3} = 'Trondheim' (string, 9 tegn lang)
- Celle-indeksering
  - V(1) = [1910] (en celle, som inneholder en tabell med tallet 1910)

Cell Arrays, fortsatt

```
>> V = {1910, 'NTNU', 'Trondheim'}
V =
    [1910]    'NTNU'    'Trondheim'
>> V{1}
ans =
        1910
>> V(1)
ans =
    [1910]
>> V(2)='UiO'
Conversion to cell from char is not possible.
>> V{2}='UiO'
V =
    [1910]    'UiO'    'Trondheim'
>>
```

Cell Arrays, fortsatt

- Pre-allokering
  - cell(<rader>, <kolonner>)
  - Oppretter en cell array med tomme elementer ([]).
- length() / size()
  - Som for alminnelige tabeller
- Referere til deler av Cell Arrays
  - Som for tabeller
  - V{1:2}
    - To første celle-verdiene i V
- celldisp()
  - Skriver ut alle elementene

```
>> V{2}='UiO'
V =
    [1910]    'UiO'    'Trondheim'
>> V{1:2}
ans =
        1910
ans =
UiO
>> celldisp(V)
V{1} =
        1910
V{2} =
UiO
V{3} =
Trondheim
>>
```

Smart: Lagring av tekststrenger

- Kan lagre tekststrenger av *ulik* lengde i samme datastruktur.

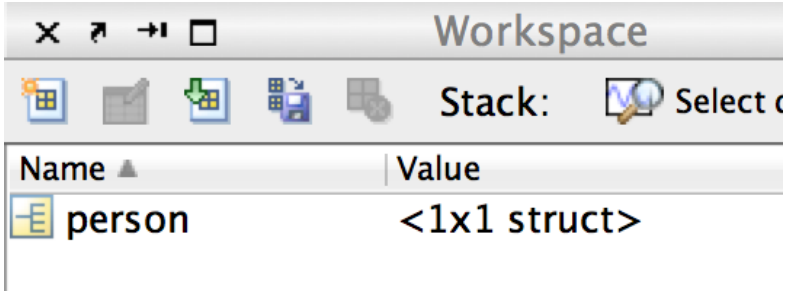
```
>> bilmerker = {'Audi', 'BMW', 'Ford', 'Toyota', 'VW'}

bilmerker =

    'Audi'    'BMW'    'Ford'    'Toyota'    'VW'

>> bilmerker{3}

ans =

Ford

>> bilmerker{3}(2)

ans =

o

>>
```

Cell Arrays, Character Sequence vs. den nye String-typen (Fra R2017a)

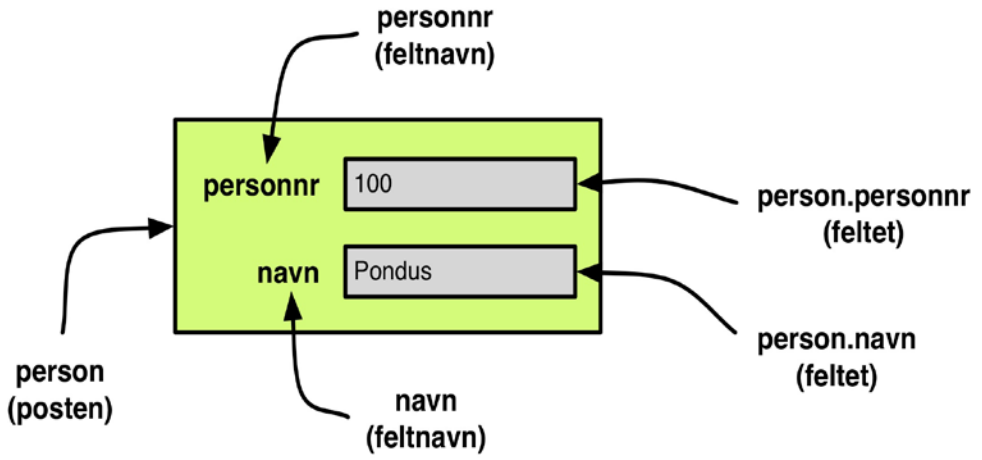
```
>> bilmerker = {'Audi', 'BMW', 'Ford', 'Toyota', 'VW'}
bilmerker =
  1×5 cell array
    {'Audi'}    {'BMW'}    {'Ford'}    {'Toyota'}    {'VW'}

>> bilmerker = ['Audi', 'BMW', 'Ford', 'Toyota', 'VW']
bilmerker =
    'AudiBMWFordToyotaVW'

>> bilmerker = ["Audi", "BMW", "Ford", "Toyota", "VW"]
bilmerker =
  1×5 string array
    "Audi"    "BMW"    "Ford"    "Toyota"    "VW"
```

Structures

Structures

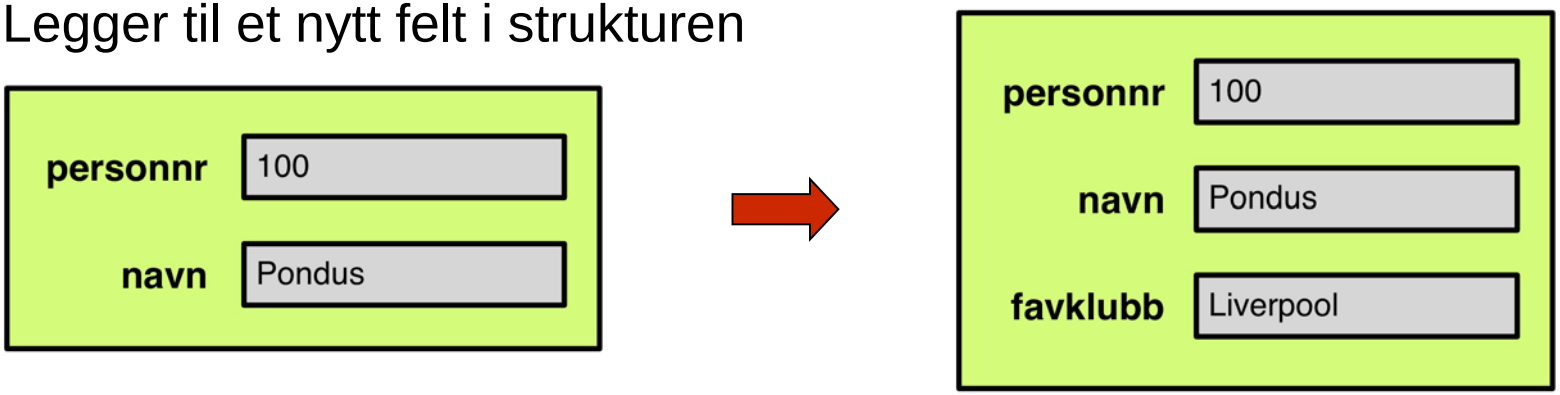


- Datastruktur som grupperer sammen verdier som hører sammen
- Navngitte felter (fields)
- person = struct('personnr',100, 'navn', 'Pondus')
- person.personnr = 100
- person.navn = 'Pondus'



Structures, forts.

- Å referere til feltene
  - <struktur-variabel>.<feltnavn> (dot operator)
  - person.favklubb = 'Liverpool'
  - Legger til et nytt felt i strukturen



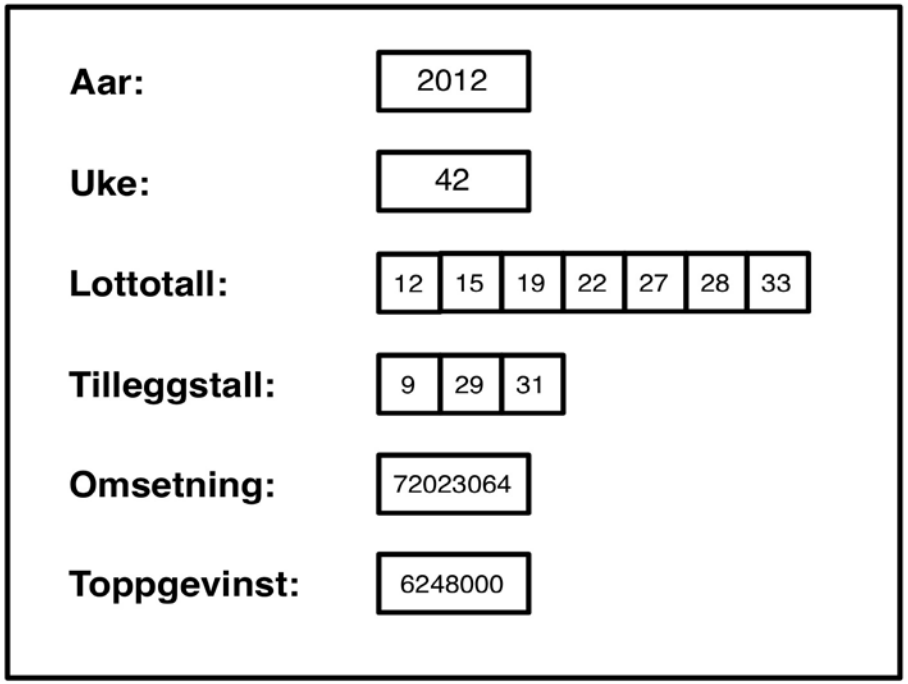
- Pre-allokering er lurt: struct(felt1, verdi1, felt2, verdi2…)
  - Begynn «bakerst» (med f.eks. struct nr. 100)
- rmfield(<struct-variabel>,<feltnavn>) fjerner alle *feltnavn*

Structures, forts.

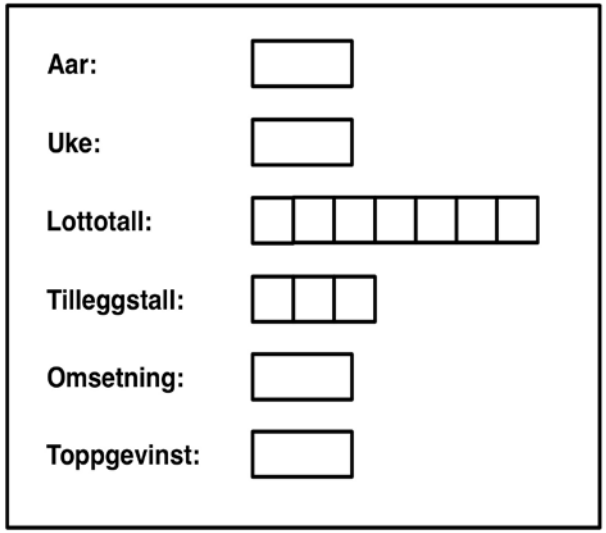
- Kan nøste strukturer inni strukturer
- Kan ha vektorer og matriser med strukturer som elementer
- Kan overføre (hele) strukturer til funksjoner
- Kan returnere strukturer fra funksjoner
- Fordeler:
  - Navngitte felt
  - Ulike datatyper
- Ulemper:
  - Kan ikke indeksere feltene i en løkke etc.
    like lett som elementene i vektorer og matriser.

(Database-) Poster

- Lottospill
- Holde data for en uke



Eksempel




```
clear, clc

lottoUke.aar = 2011;
lottoUke.uke = 41;
lottoUke.lottotall = [2 3 7 8 19 22 32];
lottoUke.ttall = [15 21 24];
lottoUke.toppgev = 3929655;
lottoUke.omsetning = 70172420;

disp(lottoUke)
disp(lottoUke.toppgev)
```

```
          aar: 2011
          uke: 41
    lottotall: [2 3 7 8 19 22 32]
        ttall: [15 21 24]
      toppgev: 3929655
    omsetning: 70172420

     3929655

>>
```

Har vi alternative datastrukturer?

- Hvilke?
- Fordeler / ulemper?

```
>> lottoRekke = cell(1,6)
lottoRekke =
    []    []    []    []    []    []
>> lottoRekke{1} = 2011
lottoRekke =
    [2011]    []    []    []    []    []
>> lottoRekke{2} = 41
lottoRekke =
    [2011]    [41]    []    []    []    []
>> lottoRekke{3} = [2 3 7 8 19 22 32]
lottoRekke =
    [2011]    [41]    [1x7 double]    []    []    []
>> lottoRekke{4} = [15 21 24]
lottoRekke =
    [2011]    [41]    [1x7 double]    [1x3 double]    []    []
>> lottoRekke{5} = 70172420
lottoRekke =
    [2011]    [41]    [1x7 double]    [1x3 double]    [70172420]    []
>> lottoRekke{6} = 3929655
lottoRekke =
  Columns 1 through 5
    [2011]    [41]    [1x7 double]    [1x3 double]    [70172420]
  Column 6
    [3929655]
>> celldisp(lottoRekke)
lottoRekke{1} =
        2011
lottoRekke{2} =
    41
lottoRekke{3} =
     2     3     7     8    19    22    32
lottoRekke{4} =
    15    21    24
lottoRekke{5} =
    70172420
lottoRekke{6} =
     3929655
>>
```

Lotto-database

- Vektor av Lotto-poster

	1	2	3	4	5
Aar:	2012	2012	2012	2012	2012
Uke:	38	39	40	41	42
Lottotall:	13 15 16 23 25 26 32	7 8 9 19 20 21 33	13 14 22 24 27 28 30	3 6 13 21 22 26 33	12 15 19 22 27 28 33
Tilleggstall:	17 21 29	2 5 23	4 25 26	16 17 30	9 29 31
Omsetning:	70707140	77286364	70914244	70928940	72023064
Toppgevinst:	12267685	2681835	2460720	3076540	6248000

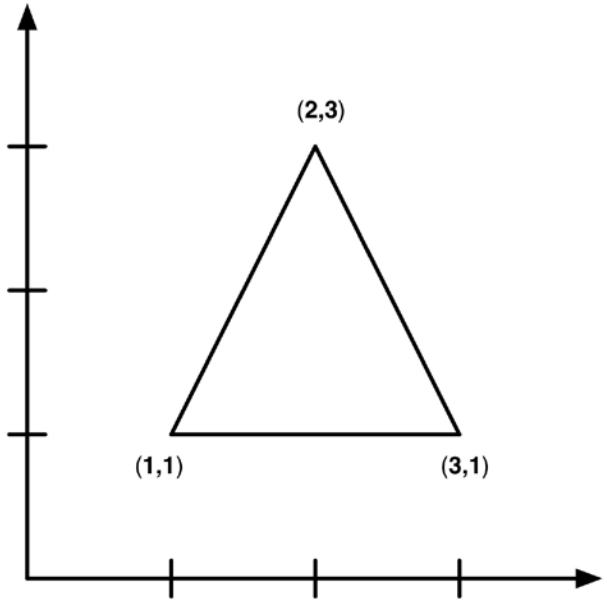
- Databasefunksjoner
  - Sett inn post, endre post, slette post
  - Maks gevinst, gjennomsnittlig omsetning, ”form-tall”, …
- Utfordringer?

Problem: Lottotrekking

- 7 tall fra 1 – 34
- Trekking *uten* tilbake-legging
- Utfordring:
  - Unngå å trekke samme tall flere ganger
- Løsninger:
  - ??

```
>> lotto

ans =

     7     9    12    17    20    21    29

>> lotto

ans =

     3    10    13    19    20    26    32

>> lotto

ans =

     2     5    16    19    20    27    32

>> lotto

ans =

     1     6    11    12    18    21    28
```

Mangekanter



- Må finne en måte å *representere* mangekanter
  - Finne egnet datastruktur
- En liste med hjørnepunkter
  - En vektor med punkt-strukturer
  - En liste med x-koordinatene og en liste med y-koordinatene
  - En matrise med x- og y-verdiene i første og andre kolonne
  - Andre forslag?

Tre alternativer

```
clear, clc

% Trekant representert med vektor med struct-er
mk_s(1) = struct('x', 1, 'y', 1);
mk_s(2) = struct('x', 2, 'y', 3);
mk_s(3) = struct('x', 3, 'y', 1);

% Trekant representert med 2 parallelle lister
xListe(1) = 1;  yListe(1) = 1;
xListe(2) = 2;  yListe(2) = 3;
xListe(3) = 3;  yListe(3) = 1;

% Trekant representert med 2-dimensjonal tabell
mk_m(1,:) = [1,1];
mk_m(2,:) = [2,3];
mk_m(3,:) = [3,1];
```

Omkrets av ”struct-mangekant”

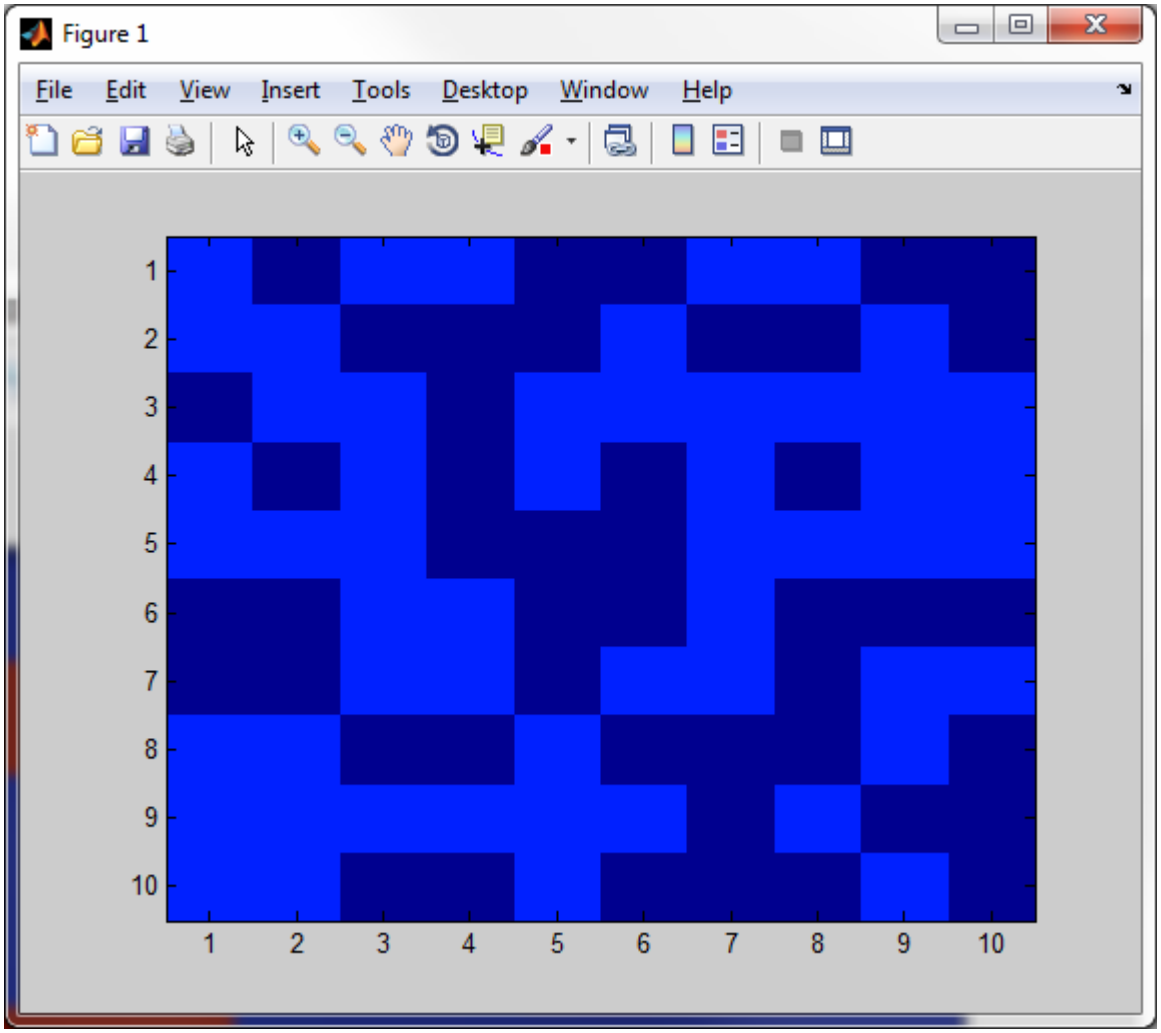
```
function omkrets = mkOmkrets_struct(mk)

antallPunkter = length(mk);

omkrets = 0;
for p = 1:1:antallPunkter
    if p < antallPunkter
        kant = sqrt((mk(p).x - mk(p+1).x)^2 +...
            (mk(p).y - mk(p+1).y)^2);
    else
        kant = sqrt((mk(p).x - mk(1).x)^2 + ...
            (mk(p).y - mk(1).y)^2);
    end % if
    omkrets = omkrets + kant;
end % for

end % function
```

- Skriv kode for de andre alternativene selv.

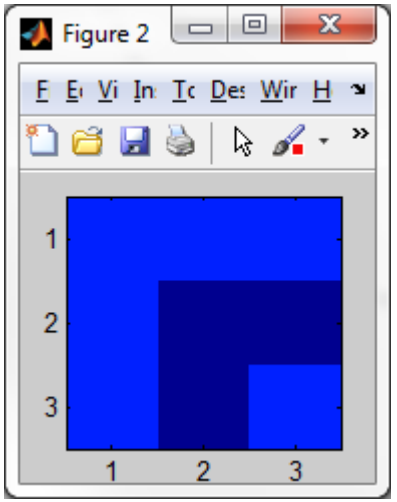
Problem: Søke etter del-tabell

- Dersom del-tabell finnes i tabell, returnere
  - true + rad og kolonne for ”øvre-venstre-hjørne”
- Ellers returneres
  - false + 0 for både rad og kolonne
- Funksjonssignatur:
  - function [finnes, rad, kol] = finnMonster(T, m)
- Problemer? Løsning?

```
>> T = randi(2,10,10) -1
T =
     1     0     1     1     0     0     1     1     0     0
     1     1     0     0     0     1     0     0     1     0
     0     1     1     0     1     1     1     1     1     1
     1     0     1     0     1     0     1     0     1     1
     1     1     1     0     0     0     1     1     1     1
     0     0     1     1     0     0     1     0     0     0
     0     0     1     1     0     1     1     0     1     1
     1     1     0     0     1     0     0     0     1     0
     1     1     1     1     1     1     0     1     0     0
     1     1     0     0     1     0     0     0     1     0

>> m = [ 1 1 1 ; 1 0 0 ; 1 0 1 ]
m =
     1     1     1
     1     0     0
     1     0     1

>> [finnes, rad, kolonne] = finnMonster(T,m)
finnes =
     1
rad =
     5
kolonne =
     7

>>
```




```
>> image(T*10)
>> image(m*10)
```



Kunnskap for en bedre verden

return og break

- Terminering av funksjoner og løkker (før tiden)
  - return

```
>> help return
 return Return to invoking function.
    return causes a return to the invoking function or to the keyboard.
    It also terminates the KEYBOARD mode.

    Normally functions return when the end of the function is reached.
    A return statement can be used to force an early return.
```

  - break

```
>> help break
 break Terminate execution of WHILE or FOR loop.
    break terminates the execution of FOR and WHILE loops.
    In nested loops, break exits from the innermost loop only.

    break is not defined outside of a FOR or WHILE loop.
    Use RETURN in this context instead.
```

return og break, forts

- Overstyrer den normale programflyten
- Gir av og til enklere kode som er lettere å forstå
- Bør brukes sparsomt
- NB! **break** avslutter bare den (innerste) løkken den står i

Eksamen 2013 og konte 2014

- Oppgave 4 (2013) er egnet for Cell Arrays
- Oppgave 4d (2014kont) er egnet for Cell Arrays