NTNU

Kunnskap for en bedre verden

Kunnskap for en bedre verden

# TDT4105 Informasjonsteknologi, grunnkurs

## MatLab:

## Filbehandling

Anders Christensen (anders@idi.ntnu.no)
Rune Sætre (satre@idi.ntnu.no)

# Læringsmål/pensum

- Filbehandling
  - Mål:
    - Forstå forskjell på tekstlig og binær form
    - Forstå hvordan data organiseres i en fil
    - Kunne overføre data til og fra filer i Matlab
  - Pensum i Matlab-boka
    - 3.6 (2.6 gammel bok) «Introduction to File Input/Output»
    - 9 Advanced File Input and Output
  - Kahoot Quiz
- Fra forrige uke: Problemløsning/programmering
  - Finne ”mønster” i større tabell

Kunnskap for en bedre verden


NTNU

# Filer: Motivasjon

- Kan ikke alltid gjenskape data eller regne ut ting på nytt
- Ofte behov for å lagre verdier som skal leses inn eller er beregnet i et program for senere bruk
- Data kan være av samfunnsmessig interesse (værvarsling) og påkrevet lagret for senere tilgang
- Mange beregninger på realistiske systemer lager store datamengder – må lagre for å kunne studere data med andre verktøy, f.eks. visualisering
- Overføre data mellom programmer

Kunnskap for en bedre verden


NTNU

# Tekstlig og binær form

- Mindre datamengder kan med fordel lagres som (ascii) tekst så vi kan se hva verdiene er
- Dette kalles å lagre data på *tekstlig form*
  - Tallverdier lagres vanligvis som binært kodede tall
    - Hvert tall (double) bruker 8 bytes i minnet (RAM)
  - Men, ved lagring på tekstlig form oversetter Matlab fra binær form til en tekststreng
    - Med tegnene: ’0’, ’1’, . . . , ’9’, ’ . ’, ’+’, ’-’, ’e’
    - Hvert enkelt tegn (enkelt-bokstav) bruker 2 bytes
- Lagring på *binær form*
  - Praktisk for datamaskiner, veldig upraktisk for mennesker!

Kunnskap for en bedre verden


NTNU

# Binærfiler versus tekstfiler

- Ulemper med tekstfiler
  - Oversetting til/fra tekstlig form
  - Tar (vanligvis) større plass
- Fordeler med tekstfiler
  - Kan lett lese innholdet (så lenge filen ikke blir alt for stor)
  - Standardisert (ASCII), kan utveksle data mellom programmer
  - Kan skrive inn nytt eller endret innhold i en teksteditor
- Filer kan bli store, veldig store
  - Data fra værberegning kan være 4 GB (4 milliarder byte)
  - Effektivitet og raske oppslag er viktig
  - Lesbarhet (for mennesker) er irrelevant
- Vi skal se på behandling av tekstfiler

Kunnskap for en bedre verden


NTNU

# Lagring av matriser

- Matlab gjør det enkelt å lagre matrisedata
  - Samme datatype
  - Regelmessig form (samme antall kolonner i alle rader)
- save <filnavn> <variabel> -ascii
  - Lager variabel i tekstfil med filnavn
- save <filnavn> <variabel> -ascii -append
  - Legger variabels innhold til i tekstfilen
- load <filnavn>
  - Henter data fra fil <filnavn> til variabel <filnavn>
- <variabel> = load('<filnavn>')
  - Henter inn data fra fil til oppgitt variabel
- type <filnavn>
  - Skriver ut innholdet i filen

Kunnskap for en bedre verden


NTNU

# Eksempel 1

```
>> v = [1 2 3 4 5];
>> save 'testfil1.txt' v -ascii;
>> type 'testfil1.txt'

   1.0000000e+00   2.0000000e+00   3.0000000e+00   4.0000000e+00   5.0000000e+00

>> save 'testfil1.txt' v -ascii -append;
>> save 'testfil1.txt' v -ascii -append;
>> save 'testfil1.txt' v -ascii -append;
>> save 'testfil1.txt' v -ascii -append;
>> type 'testfil1.txt'

   1.0000000e+00   2.0000000e+00   3.0000000e+00   4.0000000e+00   5.0000000e+00
   1.0000000e+00   2.0000000e+00   3.0000000e+00   4.0000000e+00   5.0000000e+00
   1.0000000e+00   2.0000000e+00   3.0000000e+00   4.0000000e+00   5.0000000e+00
   1.0000000e+00   2.0000000e+00   3.0000000e+00   4.0000000e+00   5.0000000e+00
   1.0000000e+00   2.0000000e+00   3.0000000e+00   4.0000000e+00   5.0000000e+00
```

Kunnskap for en bedre verden



NTNU

# Eksempel 1 (forts.)

```
>> w = load('testfil1.txt')

w =

     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
```

- save og load er den enkleste måten å bruke filer
- Forutsetter matrisedata, ikke alltid det passer

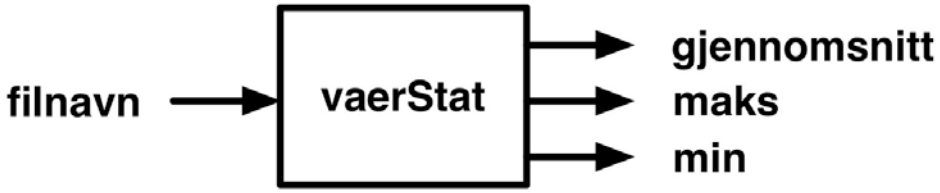
Kunnskap for en bedre verden



NTNU

# Eksempel: Værdata

- Middeltemperaturer fra Voll i september:
  - voll_sept_2011.txt
  - Hver linje: dag mnd år temp
- Ønsker funksjon som beregner:
  - Gjennomsnittstemperatur
  - Laveste temperatur
  - Høyeste temperatur




voll_sept_2011.txt

```
1 9 2011 12.4
2 9 2011 13.1
3 9 2011 14.5
4 9 2011 16.8
5 9 2011 15.5
6 9 2011 14.7
7 9 2011 14.1
8 9 2011 11.2
9 9 2011 10.7
10 9 2011 9.5
11 9 2011 14.1
12 9 2011 14.3
13 9 2011 12.3
14 9 2011 10.6
15 9 2011 8.5
16 9 2011 8.1
17 9 2011 11.2
18 9 2011 10.4
19 9 2011 11.3
20 9 2011 9.5
21 9 2011 10.1
22 9 2011 8.5
23 9 2011 10.0
24 9 2011 10.3
25 9 2011 10.7
26 9 2011 10.7
27 9 2011 9.0
28 9 2011 11.4
29 9 2011 16.1
30 9 2011 14.1
```

Kunnskap for en bedre verden

# Eksempelkjøring

```
>> [Te Mi Ma] = vaerStat('voll_sept_2011.txt')
Te =
   11.789999999999999
Mi =
    8.100000000000000
Ma =
   16.800000000000001
>>
```

Kunnskap for en bedre verden



NTNU

# Lavnivå filbehandling

- Åpner filer (etablerer kobling til filen)
  - <fil-id> = fopen('<filnavn>', '<tilgangstype>');
  - fil-id er en *filpeker* (referanse til filen)
  - fil-id settes til -1 ved problemer
  - Tilgangstyper:
    - r – lese (fra starten)
    - w – skrive (sletter gammelt fil-innhold først)
    - a – legge til (skrive på slutten av filen)
- Lukker filen når vi er ferdig med den
  - <status> = fclose(<fil-id>)
  - Returnerer 0 når det går bra, -1 ellers

Kunnskap for en bedre verden


NTNU

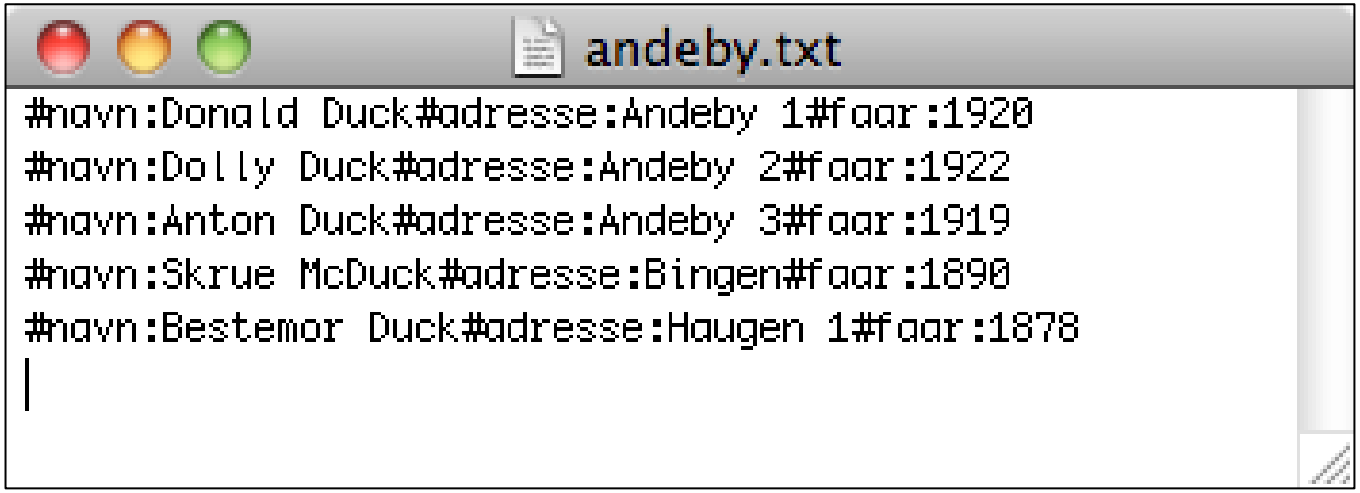
# Lesing/skriving

- fgetl(<fil-id>)
  - Leser inn en linje
  - Returnerer en tekststreng med linjens innhold
- feof(<fil-id>)
  - *usann* så lenge det er mer data igjen i filen (som ikke er lest)
  - *sann* når vi har kommet til slutten av filen
- fprintf(<fil-id>, <formatstreng>, <verdier>)
  - Skriver til filen med <fil-id>
  - Som til skjerm
- Mange flere (spesialiserte) muligheter, se læreboka og hjelp i Matlab.

Kunnskap for en bedre verden


NTNU

# Eksempelfil: Personregister




- Data om en person på hver linje
- Feltindikatorer: #navn: #adresse: #faar:
- Leser inn en og en linje og plukker ut data

Kunnskap for en bedre verden

# lesAndebyFil.m

```
clear, clc

% aapner filen
fid = fopen('andeby.txt', 'r');

% sjekket at aapningen gikk bra
if fid == -1
    disp('Kunne ikke aapne filen')
else
    % leser alle linjene
    while ~feof(fid)
        % leser en linje
        filLinje = fgetl(fid);
        % skriver ut linjen
        fprintf('%s\n', filLinje)
    end % while

    lukkStatus = fclose(fid);
    if lukkStatus == 0
        disp('Lukket filen')
    else
        disp('Kunne ikke lukke filen')
    end
end
```

Kunnskap for en bedre verden



NTNU

# Eksempelkjøring

```
#navn:Donald Duck#adresse:Andeby 1#faar:1920
#navn:Dolly Duck#adresse:Andeby 2#faar:1922
#navn:Anton Duck#adresse:Andeby 3#faar:1919
#navn:Skrue McDuck#adresse:Bingen#faar:1890
#navn:Bestemor Duck#adresse:Haugen 1#faar:1878
Lukket filen
>>
```

- Leser en og en linje fra filen
- Skriver ut hele linjen

Kunnskap for en bedre verden

# Plukker ut data: lesAndebyFil2.m

```
% leser alle linjene
while ~feof(fid)
    % leser en linje
    filLinje = fgetl(fid);
    % finner #-ene
    navnStart = 1;
    adresseStart = strfind(filLinje, '#adresse:');
    faarStart= strfind(filLinje, '#faar:');
    % finner linjelengden
    linjeLengde = length(filLinje);
    % plukker ut data
    navn = filLinje(7:adresseStart-1);
    adresse = filLinje(adresseStart+9:faarStart-1);
    faar = filLinje(faarStart+6:linjeLengde);
    fprintf('Navn:    %s\n', navn)
    fprintf('Adresse: %s\n', adresse)
    fprintf('Faar:    %s\n', faar)
end % while

lukkStatus = fclose(fid);
if lukkStatus == 0
    disp('Lukket filen')
else
    disp('Kunne ikke lukke filen')
end
```

Kunnskap for en bedre verden

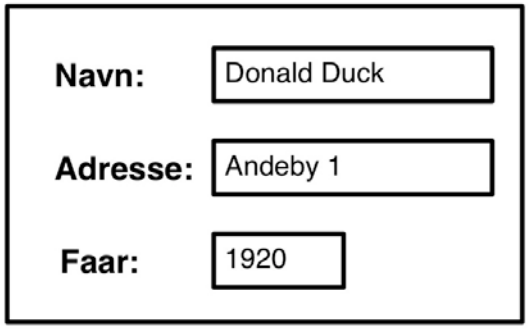
# Kjøreeksempel

```
Navn:     Donald Duck
Adresse:  Andeby 1
Faar:     1920
Navn:     Dolly Duck
Adresse:  Andeby 2
Faar:     1922
Navn:     Anton Duck
Adresse:  Andeby 3
Faar:     1919
Navn:     Skrue McDuck
Adresse:  Bingen
Faar:     1890
Navn:     Bestemor Duck
Adresse:  Haugen 1
Faar:     1878
Lukket filen
```

Kunnskap for en bedre verden

# Andeby-”database”

- Leser inn persondata til en vektor av strukturer
- Personpost (struktur):




- Vektor av personer:

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Navn: Donald Duck | Navn: Dolly Duck | Navn: Anton Duck | Navn: Skrue McDuck | Navn: Bestemor Duck |
| Adresse: Andeby 1 | Adresse: Andeby 2 | Adresse: Andeby 3 | Adresse: Bingen | Adresse: Haugen 1 |
| Faar: 1920 | Faar: 1922 | Faar: 1919 | Faar: 1890 | Faar: 1978 |

Kunnskap for en bedre verden



NTNU

# lesAndeby.m

```
function data = lesAndeby
% leser persondata fra fil til vektor

   % aapner filen
   fid = fopen('andeby.txt', 'r');

   % sjekket at aapningen gikk bra
   if fid == -1
      exit('Feil i lesAndeby: Kunne ikke aapne filen')
   else
      % leser alle linjene
      personNr = 1;
      while ~feof(fid)
         % leser en linje
         filLinje = fgetl(fid);
         % finner #-ene
         navnStart = 1;
         adresseStart = strfind(filLinje, '#adresse:');
         faarStart= strfind(filLinje, '#faar:');
         % finner linjelengden
         linjeLengde = length(filLinje);
         % plukker ut data
         data(personNr).navn = filLinje(7:adresseStart-1);
         data(personNr).adresse = filLinje(adresseStart+9:faarStart-1);
         data(personNr).faar = str2num( filLinje(faarStart+6:linjeLengde) );
         % neste personnr
         personNr = personNr + 1;
    end % while

    lukkStatus = fclose(fid);
    if lukkStatus == 0
        disp('Lukket filen')
    else
        disp('Kunne ikke lukke filen')
    end
end
```

Kunnskap for en bedre verden

# Kjøreeksempel

```
>> dataVektor = lesAndeby;
Lukket filen
>> dataVektor(1)

ans =

       navn: 'Donald Duck'
    adresse: 'Andeby 1'
       faar: 1920

>> dataVektor(5).navn

ans =

Bestemor Duck

>>
```

Kunnskap for en bedre verden

# Å legge til en «person»

- Lager funksjon som registrerer ny Andeby-innbygger
- Tar person-post som inn-parameter
- Åpner filen for å legge til (‘a’)
- Lager tekstlinje ut fra person-posten
- Skriver tekstlinje til fil
  - fprintf(fid, '%s\n', linje);
- Lukker filen

Kunnskap for en bedre verden


NTNU

# regAndebyKarakter.m



```
function regAndebyKarakter(karakter)
% tar inn karakter og legger til i datafilen

    % aapner filen for tillegg av data
    fid = fopen('andeby.txt', 'a');

    % sjekket at aapningen gikk bra
    if fid == -1
        exit('Feil i lesAndeby: Kunne ikke aapne filen')
    else
        linje = ['#navn:' karakter.navn];
        linje = [linje '#adresse:' karakter.adresse];
        linje = [linje '#faar:' num2str(karakter.faar) ];
        fprintf(fid, '%s\n', linje);
    end

    % lukker filer
    lukkStatus = fclose(fid);
    if lukkStatus == 0
        disp('Lukket filen')
    else
        disp('Kunne ikke lukke filen')
    end

end % function
```

# Kjøreeksempel

```
>> p.navn = 'Guffen';
>> p.adresse = 'Haugen 1';
>> p.faar = 1895;
>> regAndebyKarakter(p)
Lukket filen
>> lesAndebyFil
#navn:Donald Duck#adresse:Andeby 1#faar:1920
#navn:Dolly Duck#adresse:Andeby 2#faar:1922
#navn:Anton Duck#adresse:Andeby 3#faar:1919
#navn:Skrue McDuck#adresse:Bingen#faar:1890
#navn:Bestemor Duck#adresse:Haugen 1#faar:1878
#navn:Guffen#adresse:Haugen 1#faar:1895
Lukket filen
>>
```

Kunnskap for en bedre verden

# Problemet fra forrige uke: Søke etter del-tabell

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

Kunnskap for en bedre verden

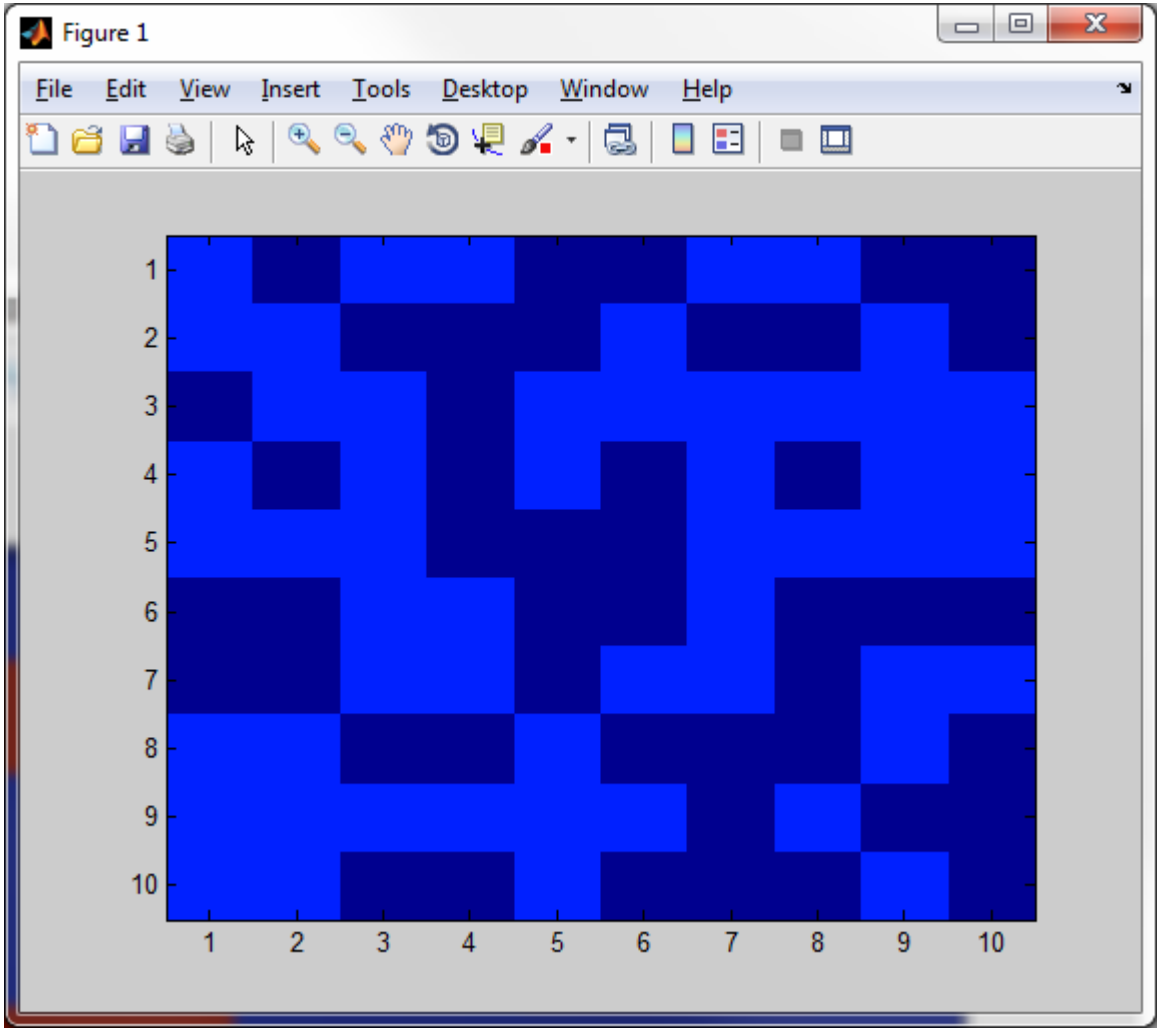

NTNU

```
>> 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

>>
```

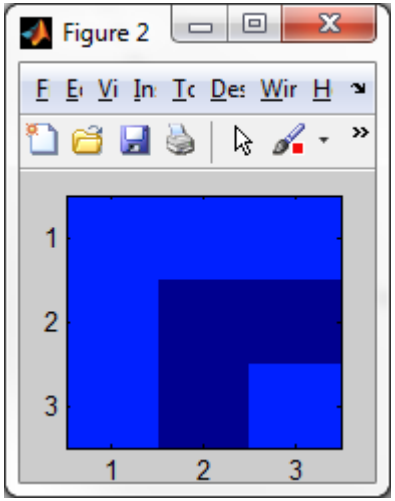
Kunnskap for en bedre verden



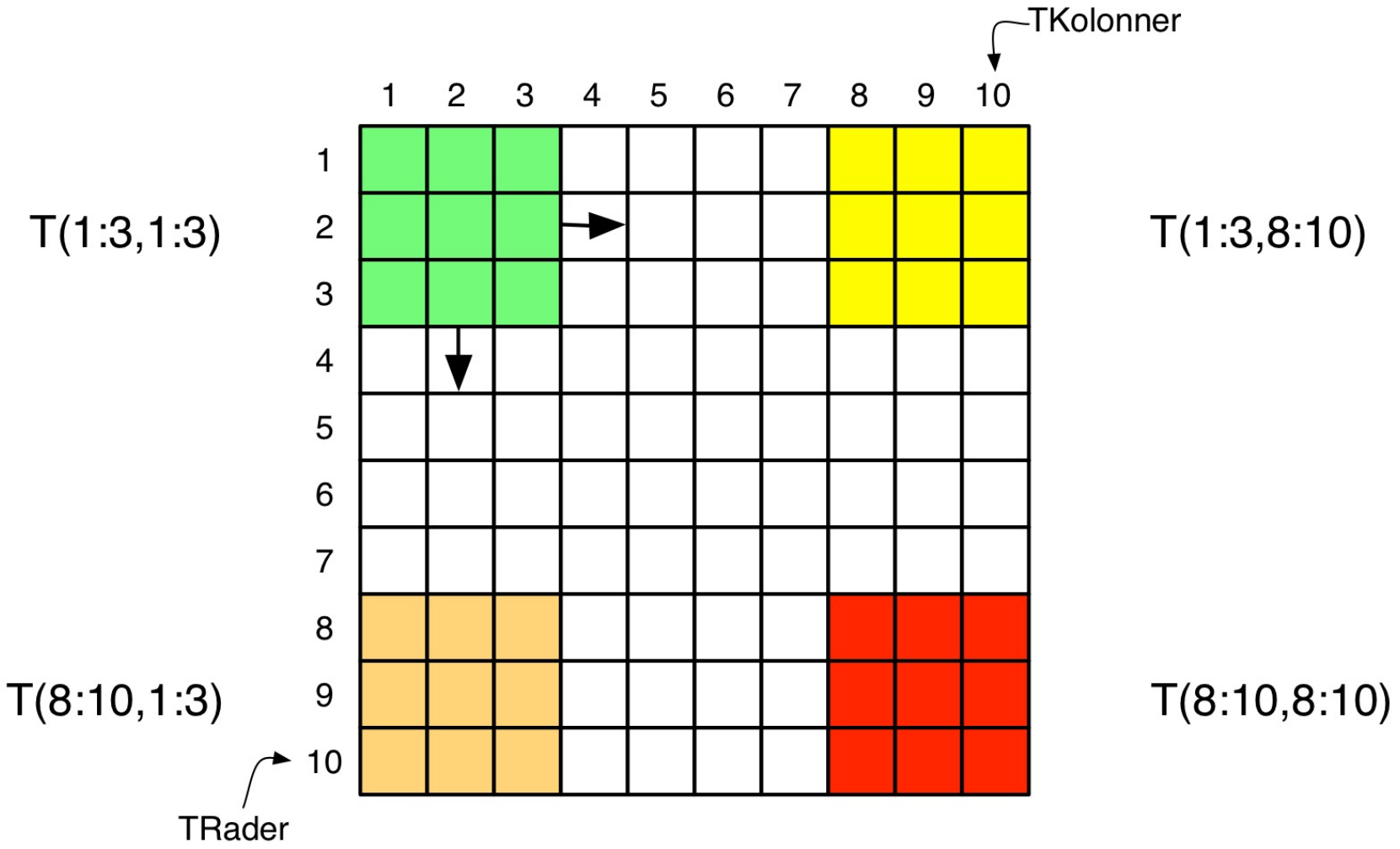
NTNU




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




Kunnskap for en bedre verden




Kunnskap for en bedre verden

```
function [finnes, rad, kol] = finnMonster(T, m)
% Leter etter m i T, returnerer indeks til første forekomst

    [TRader, TKolonner] = size(T);
    [mRader, mKolonner] = size(m);

    % sjekker om mønster er større enn tabellen
    if (mRader > TRader) || (mKolonner > TKolonner)
        finnes = false;
        rad = 0;
        kol = 0;
        return
    end % if

    % leter ettter mønster
    for rad=1:1:(TRader-mRader+1)
        for kol=1:1:(TKolonner-mKolonner+1)

            if isequal(m, T(rad:(rad+mRader-1), kol:(kol+mKolonner-1)))
                finnes = true;
                return
            end % if

        end % for
    end % for

    % fant ikke mønster
    finnes = false;
    rad = 0;
    kol = 0;

end %function
```

Kunnskap for en bedre verden



NTNU

# 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.
```

Kunnskap for en bedre verden

# 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

Kunnskap for en bedre verden



NTNU