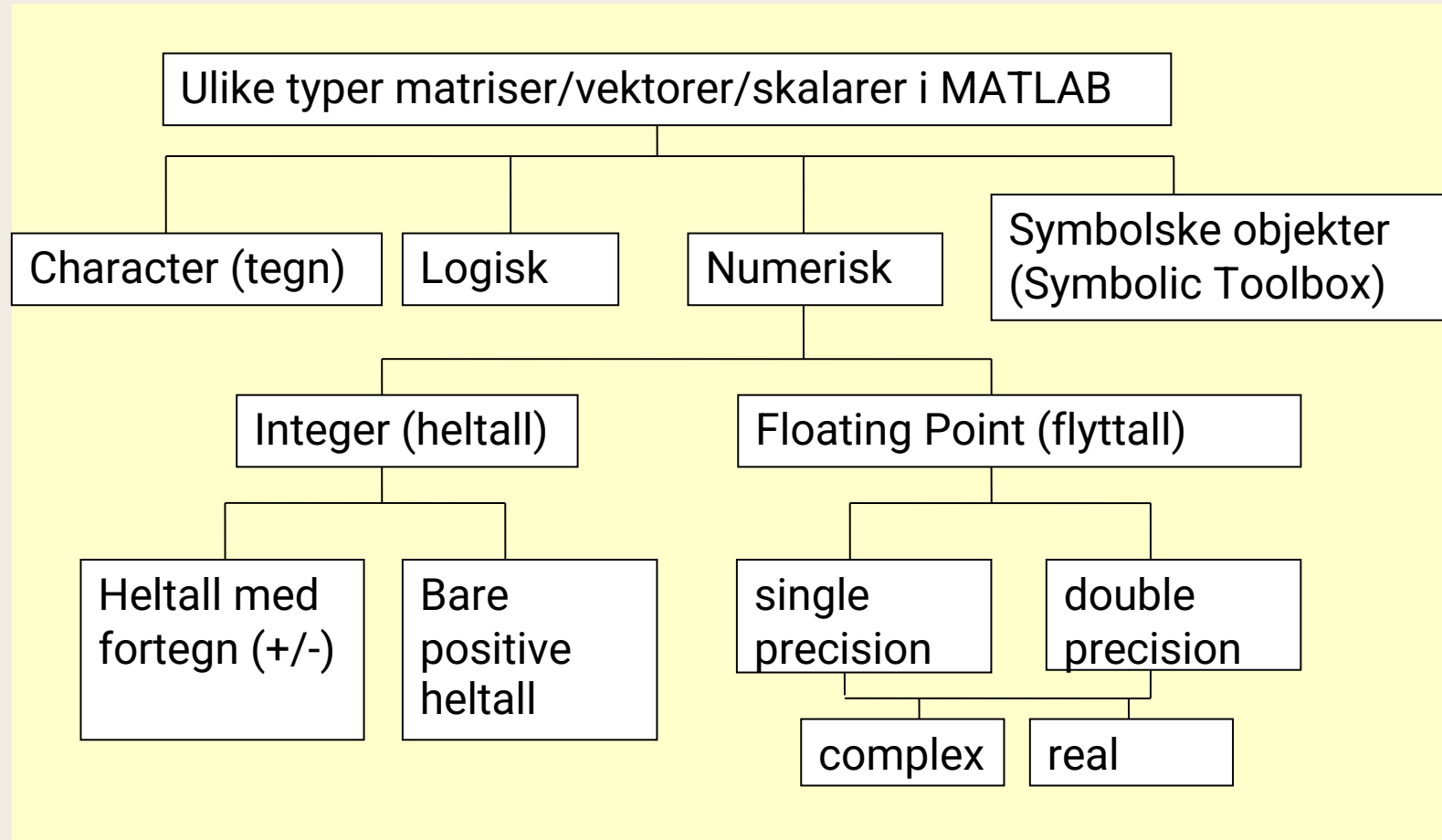


Notat 1, del 8 (av totalt 9 deler)

Dat typer og «samle»-typer i MATLAB

Ulike datatyper i MATLAB



Numeriske datatyper

- Flyttall (kommatall) med dobbel presisjon
- Flyttall (kommatall) med enkel presisjon
- Heltall med fortegn (+/-)
- Heltall uten fortegn (bare positive)

Heltall

- Det finnes 8 heltallstyper
- Antall bit per tall varierer

MATLAB Integer Types			
8-bit signed integer	int8	8-bit unsigned integer	uint8
16-bit signed integer	int16	16-bit unsigned integer	uint16
32-bit signed integer	int32	32-bit unsigned integer	uint32
64-bit signed integer	int64	64-bit unsigned integer	uint64

Husk: 8 bit = 1 byte

Tallområdet for heltallstypene

- uint8: [0,255] ($2^8 = 256$ forskjellige tall)
- int8: [-128,127] ($2^8 = 256$ forskjellige tall)
- uint16: [0,65535] ($2^{16} = 65536$ forskjellige tall)
- int16: [-32768,32767] ($2^{16} = 65536$ forskjellige tall)

Prøv **intmax** og **intmin** på de andre typene:

- uint32
- int32
- uint64
- int64

Hvorfor bruke heltallstyper?

- Eksempel:
- Kan brukes til å lagre bilder: Store matriser med tallverdier i området $[0,255]$ for rød, grønn og blå intensitet i hvert piksel.
- Dersom `uint` brukes, kan lagringsmengden reduseres drastisk:

Fra 8 byte pr. farge pr. piksel til 1 byte.

Komplekse tall

Bruker dobbelt så stor plass som et reelt tall, siden både reell og imaginær del tar hver sin “double” (som standard).

Eks.:

```
a = 5 + 3i;
```

Variabelen **a** inneholder nå et komplekst tall.

Tegn og tekst

- En tekststreng er en vektor med char-verdier
- **char** er datatypen for et tegn
- “char” er en forkortelse for “character” = tegn
- 1-4 byte for hver char-verdi (med UTF-8)

Workspace

Name	Value	Size	Bytes	Class
H	'Holly'	1x5	10	char
ans	'y'	1x1	2	char

Workspace Current Directory

Command History

```
clear,clc  
intmax('int8')  
intmax('uint8')  
intmin('int8')  
intmin('uint8')  
clear,clc  
H='Holly'  
H(5)
```

Command Window

```
>> H='Holly'  
H =  
Holly  
>> H(5)  
ans =  
y  
>>
```

Det femte elementet i vektoren H er bokstaven 'y'

9

OVR

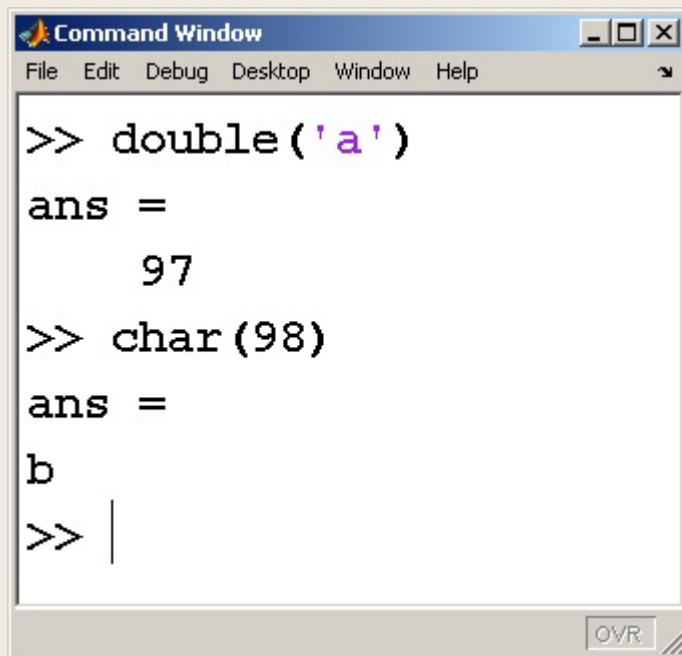
ASCII-tabell (erstattet av Unicode)

- Alle tegn (og tall) lagres i datamaskinen som en samling av 0-ere og 1-ere.
- Hvert char-tegn lagres ved bruk av et bestemt antall bit
- **ASCII- eller Unicode-tegnsettet** definerer kombinasjonen numeriske koder for det enkelte tegn.
- Hvert tegn har en bestemt plass i tegnsettet
 - ‘a’ (latinsk a) har plass 97
 - ‘α’ (gresk alfa) har plass 945
 - ‘א’ (hebraisk alef) har plass 1488

Utdrag fra ASCII-tabellen

Binært	Desimalt	Heksadesimalt	Tegn
110 0001	97	61	a
110 0010	98	62	b
110 0011	99	63	c
110 0100	100	64	d
110 0101	101	65	e
110 0110	102	66	f
110 0111	103	67	g
110 1000	104	68	h
110 1001	105	69	i
110 1010	106	6A	j
110 1011	107	6B	k
110 1100	108	6C	l
1101101	109	6D	m

Konvertere mellom datatyper



```
Command Window
File Edit Debug Desktop Window Help
>> double('a')
ans =
    97
>> char(98)
ans =
b
>> |
```

Vi kan konvertere mellom tegn og og tall

Bruker **char** til å konvertere fra tall til tegn

```
>> [char(5803) char(5850) char(5794)]
ans =
' Æ Å '
```

Symbolisk datatype

- “symbolic toolbox” bruker symbolisk datatype for å kunne gjøre symbolisk algebra
- En symbolisk variabel opprettes ved å bruke funksjonen **sym**
- Mer om dette i del 8

Logisk datatype

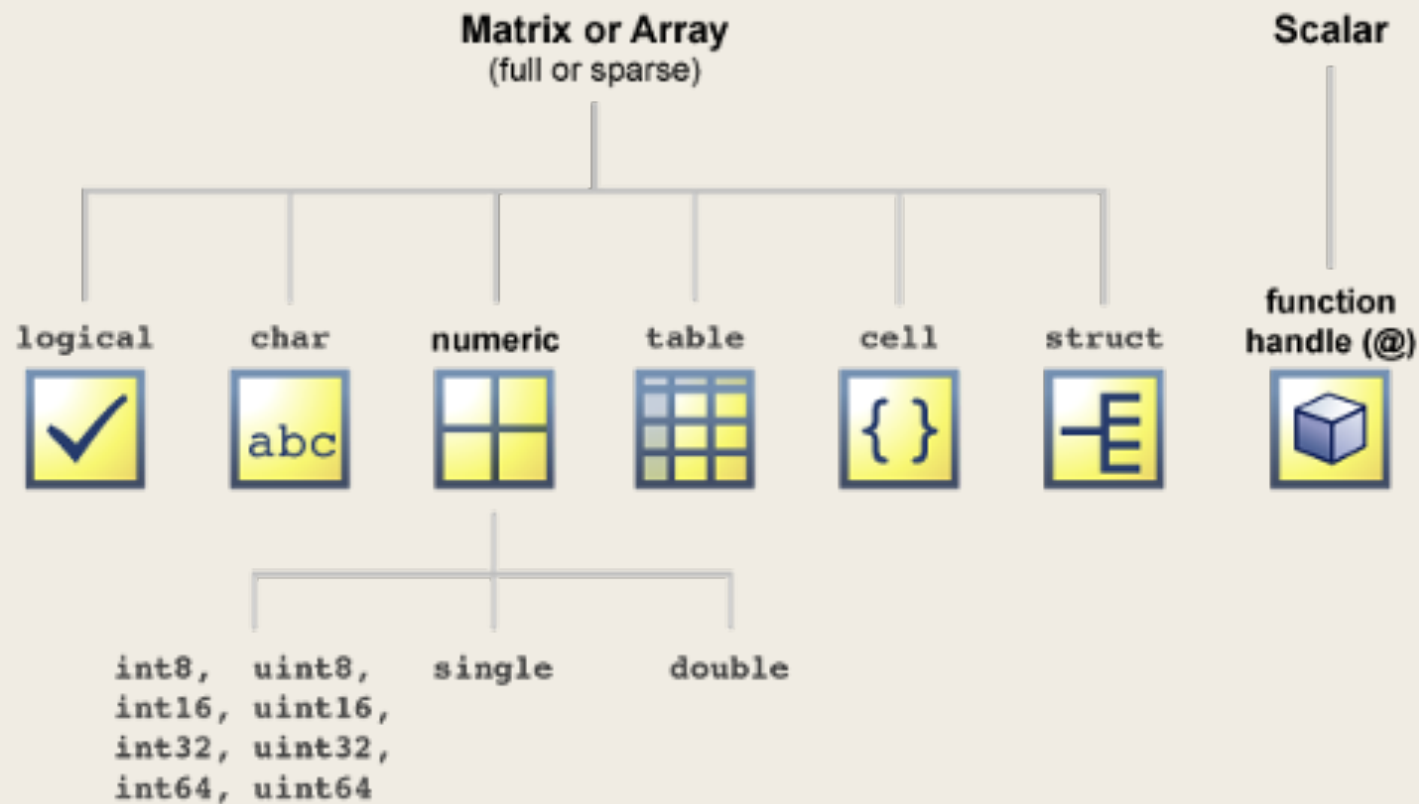
Merk at Matlab bruker små bokstaver
true og false

mens Python bruker stor forbokstav,
True og False

- Logiske data kan bare ha en av to verdier
 - Sann
 - Usann
- MATLAB bruker
 - 0 for usann
 - 1 for sann

```
Command Window
>> x=1:5;
>> y=[2,0,2,9,4];
>> z=x>y
z =
     0     1     1     0     1
>> find(x>y)
ans =
     2     3     5
fx >>
```

«Samle»-typer



«Samle»-typen **cell**

- En **cell** kan inneholde flere matriser, vektorer og/eller elementer av ulik størrelse og/eller datatype.
- **cell**-typen er fin å ha når vi ønsker å samle ulike datavariabler som naturlig hører sammen i en og samme “sekk”.
- Vi bruker krøllparanteser, { og }, når en cell skal lages:

Eks.:

```
>> C = {[1 2 3], 'abcde', 3.58}
```

```
C =
```

```
[1x3 double] 'abcde' [3.5800]
```


Current ...

Workspace

Command Window

```
>> A = 1:3;  
>> B = ['abcdefg'];  
>> C = single([1 2 3 ;4 5 6]);  
fx >> |
```

Tre variabler med ulik størrelse og av ulik datatype

Name	Value	Size	Bytes	Class
A	[1,2,3]	1x3	24	double
B	'abcdefg'	1x7	14	char
C	[1,2,3;4,...	2x3	24	single

Command History

```
clear,clc  
some_data =[1 2 3;4 5 6;7 8  
save some_data  
clear,clc  
load some_data  
clear,clc  
A = 1:3;  
B = ['abcdefg'];  
C = single([1 2 3 ;4 5 6]);
```

Current ...

Command Window

```
>> A = 1:3;  
>> B = ['abcdefg'];  
>> C = single([1 2 3 ;4 5 6]);  
>> my_cellarray = {A,B,C}  
my_cellarray =  
    Columns 1 through 2  
    [1x3 double]      'abcdefg'  
    Column 3  
    [2x3 single]
```

fx >> |

Lager en cell

Workspace

Name	Value	Size	Bytes	Class
A	[1,2,3]	1x3	24	double
B	'abcdefg'	1x7	14	char
C	[1,2,3;4,...	2x3	24	single
my_c...	<1x3 ce...	1x3	242	cell

Command History

```
some_data = [1 2 3;4 5 6;7 8  
save some_data  
clear,clc  
load some_data  
clear,clc  
A = 1:3;  
B = ['abcdefg'];  
C = single([1 2 3 ;4 5 6]);  
my_cellarray = {A,B,C}
```

Start

OVR

Indeksering

- Det er to måter å hente ut data/informasjon fra en celletabell på:
- Vanlige paranteser ()
 - Vi får en **delmengde** av innholdet i celletabellen
- Krøllparanteser { }
 - Vi får **dataene/verdien** som er lagret i celletabellen

```
Command Window
>> my_cellarray(1)
ans =
    [1x3 double]
>> my_cellarray{1}
ans =
     1     2     3
fx >> |
```

```
Command Window
>> my_cellarray(1)
ans =
    [1x3 double]
>> my_cellarray{1}
ans =
     1     2     3
>> my_cellarray{3}
ans =
     1     2     3
     4     5     6
>> my_cellarray{3} (1,2)
ans =
     2
fx >>
```

For å hente ut deler av en vektor/matrise som ligger i en celle, må både krøllparanteser og vanlige paranteser brukes.

«Sample»-typen struct

- **struct**-typen ligner litt på **cell**-typen
- Flere ulike variabler kan samles i en “sekk”
- I stedet for å bruke indeksering, har hver variabel et *navn*. Vi kaller hver variabel et *felt* (“field”)
- Vi lager en struktur ved å bruke . (punktum) mellom struktur-navnet og et nytt feltnavn

MATLAB 7.12.0 (R2011a)

File Edit Debug Desktop Window Help

C:\Users\Holly\Documents\MATLAB

Shortcuts How to Add What's New

Current ...

Command Window

```
>> A = 1:3;
>> B = ['abcdefg'];
>> C = single([1 2 3 ;4 5 6]);
>> my_structure.some_numbers = A
my_structure =
    some_numbers: [1 2 3]
>> my_structure.some_letters = B
my_structure =
    some_numbers: [1 2 3]
    some_letters: 'abcdefg'
>> my_structure.some_more_numbers = C
my_structure =
    some_numbers: [1 2 3]
    some_letters: 'abcdefg'
    some_more_numbers: [2x3 single]
```

fx >> |

Workspace

Name	Value	Size	Bytes	Class
A	[1,2,3]	1x3	24	double
B	'abcdefg'	1x7	14	char
C	[1,2,3;4,5,6]	2x3	24	single
my_st...	<1x1 str...	1x1	434	struct

Command History

```
my_cellarray{3}
my_cellarray{3} (1
clear,clc
A = 1:3;
B = ['abcdefg'];
C = single([1 2 3
    ructure.some
    ructure.some
my_structure.some
```

Strukturer my_structure har 3 felt

The image shows a MATLAB interface with three main windows: a file explorer on the left, a Command Window in the center, and a Workspace window on the right.

File Explorer: Shows a list of files including `hybrid_compar...`, `Jacobi.m`, `lake_powell.dat`, `log_plotting_ex...`, `log_plotting_ex...`, `Logarithmic_rel...`, `mandelbrot.m`, `matlab.mat`, `mimas.jpg`, `motion.m`, `my_data1.mat`, `my_data2.mat`, `my_data3.mat`, `my_dot.m`, `my_example_fil...`, `my_file.txt`, `my_function.m`, `my_neat_matla...`, `my_output_file....`, `newstats.m`, `num_grains.m`, `pnorm.m`, and `pnorm1.m`.

Command Window: Contains the following MATLAB code:

```
>> my_structure(2).some_numbers = [2 4 6 8]
my_structure =
1x2 struct array with fields:
    some_numbers
    some_letters
    some_more_numbers
```

Workspace: Displays a table of variables in the workspace:

Name	Value	Size	Bytes	Class
A	[1,2,3]	1x3	24	double
B	'abcdefg'	1x7	14	char
C	[1,2,3,4,...]	2x3	24	single
my_st...	<1x2 str...	1x2	534	struct

A red circle highlights the `my_st...` entry in the Workspace window.

Text Overlay: A pink box contains the text: "Vi kan legge mer data inn i strukturen etterpå, ved å definere strukturnavn og feltnavn der dataene skal inn".

Editor Window: Shows the following MATLAB code:

```
le([1 2 3
B = ['abcdefg'];
my_structure.some_
my_structure.some_
my_structure.some_
clc
my_structure(2).so
```

Current ...

<< M... >>

Name ▲

my_data1.mat
my_data2.mat
my_data3.mat
my_dot.m
my_example_fil...
my_file.txt
my_function.m
my_neat_matla...
my_output_file....
newstats.m
num_grains.m
pnorm.m
pnorm1.m

Details ▼

Select a file to view details

Command Window

>> my_structure
my_structure =
1x2 struct array with fields:
 some_numbers
 some_letters
 some_more_numbers

fx >>

Workspace

Select data t...

Name ▲	Value	Size	Bytes	Class
A	[1,2,3]	1x3	24	double
B	'abcdefg'	1x7	14	char
C	[1,2,3;4,...	2x3	24	single
my_st...	<1x2 str...	1x2	534	struct

[1 2 3

B = ['abcde'fg'];
my_structure.some_
my_structure.some_
my_structure.some_
clc
my_structure(2).so
clc
my_structure

Hente ut beskrivelse om strukturen

Current ...

Command Window

```
>> my_structure(1)
ans =

    some_numbers: [1 2 3]
    some_letters: 'abcdefg'
    some_more_numbers: [2x3 single]
>> my_structure(2)
ans =

    some_numbers: [2 4 6 8]
    some_letters: []
    some_more_numbers: []
fx >>
```

Workspace

Name	Value	Size	Bytes	Class
A	[1,2,3]	1x3	24	double
B	'abcdefg'	1x7	14	char
C	[1,2,3;4,...	2x3	24	single
ans	<1x1 str...	1x1	292	struct
my_st...	<1x2 str...	1x2	534	struct

Command History

```
my_structure.some_
my_structure.some_
clc
my_structure
my_structure(1)
my_structure(2)
```

Hente ut beskrivelse om hvert felt

Start

OVR

Universitetet i Stavanger

Current ...

Command Window

```
>> my_structure(2).some_numbers(2)
ans =
    4
fx >>
```

Hente ut verdier fra et bestemt felt i strukturen

Workspace

Name	Value	Size	Bytes	Class
A	[1,2,3]	1x3	24	double
B	'abcdefg'	1x7	14	char
C	[1,2,3;4,...	2x3	24	single
ans	4	1x1	8	double
my_st...	<1x2 str...	1x2	534	struct

Command History

```
clc
my_structure(1)
my_structure(2)
clc
my_structure.some_
clc
my_structure(2).so
clc
my_structure(2).so
```

Start

OVR

Konvertering mellom typer (eksempler)

- table2array
 - struct2table
 - mat2cell
 - cell2mat
 - array2table
 - num2str
 - hex2dec
-
- Disse går begge veier. Eks num2str og str2num
 - Skriv f.eks. >> table2 og trykk Tab for å se komplett liste