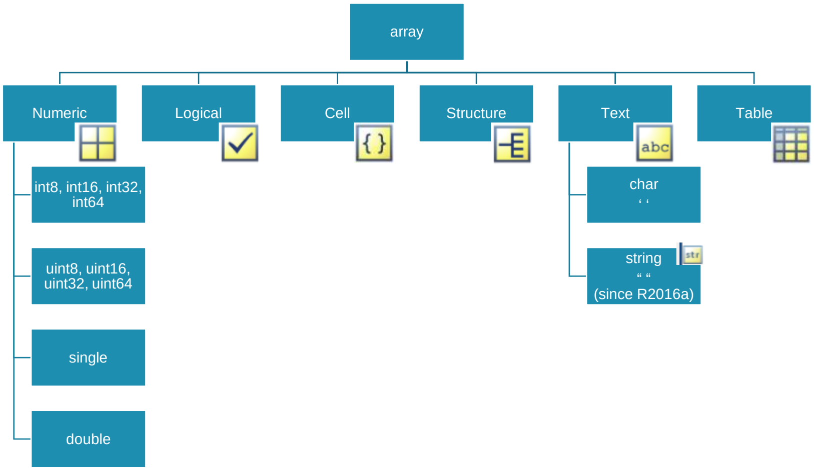
MATLAB

Fundamental Data Types: more

1

Overview data types (classes)



array

Numeric

- int8, int16, int32, int64
- uint8, uint16, uint32, uint64
- single
- double

Logical

Cell

Structure

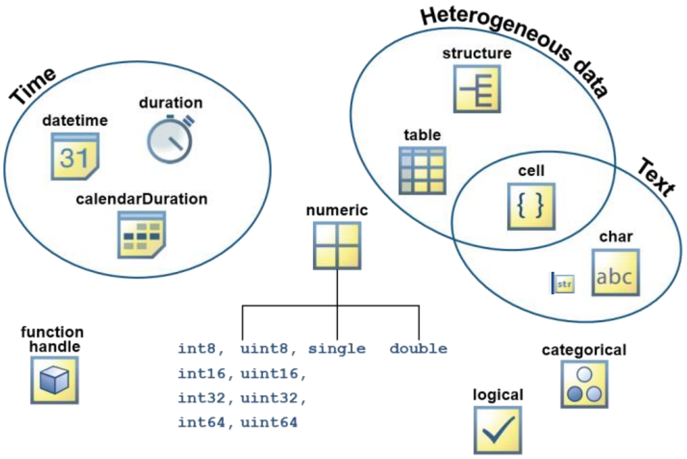
Text

- char ' '
- string " " (since R2016a)

Table

https://nl.mathworks.com/help/matlab/matlab_prog/fundamental-matlab-classes.html

3



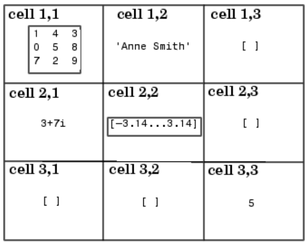
Taken from MATLABacademy

Cell array

Class Name	Documentation	Intended Use
`cell`	Cell Arrays	• Cells store arrays of varying classes and sizes. • Allows freedom to package data as you want. • Manipulation of elements is similar to numeric or logical arrays. • Method of passing function arguments. • Use in comma-separated lists. • More memory required for overhead

Cell Array

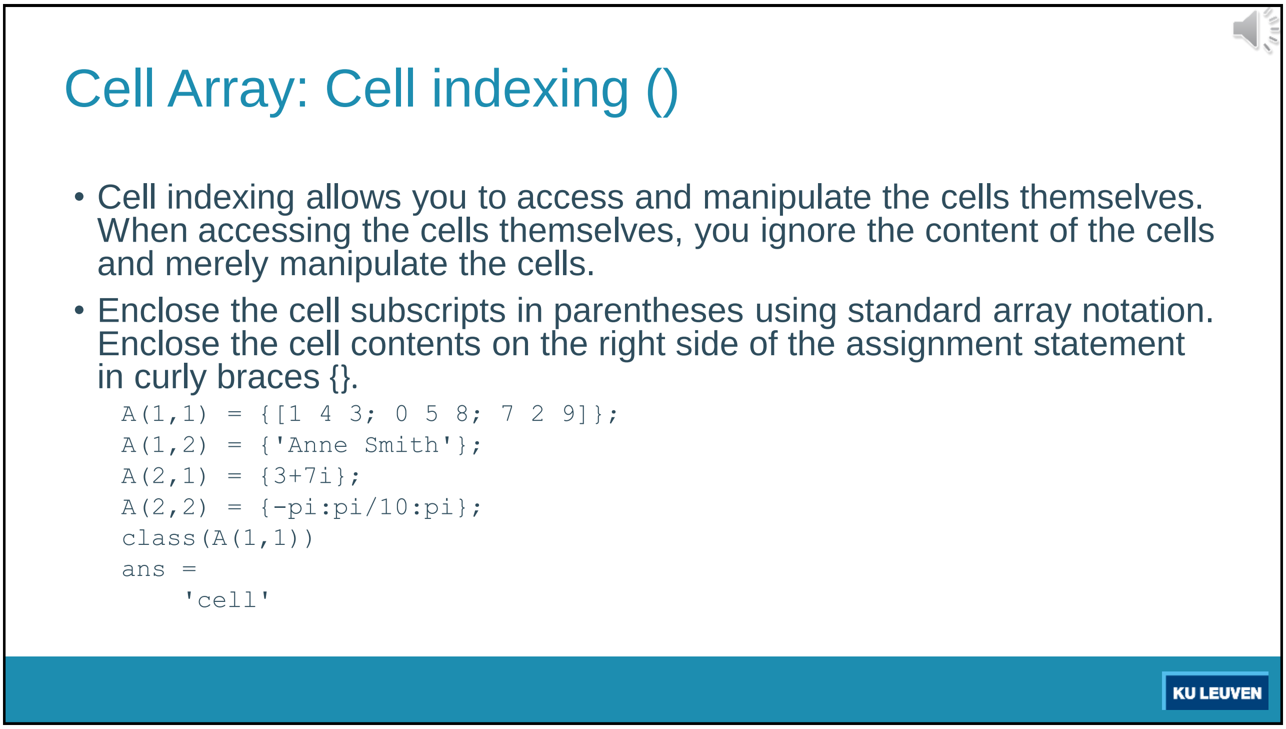
- Most *general* MATLAB data structure: ‘*spreadsheet*’
- Provides a storage mechanism for *dissimilar* kinds of data, for any type of data.
- A cell array is just like a matrix except each entry can be any data type, not just a number.



cell 1,1 1 4 3 0 5 8 7 2 9	cell 1,2 'Anne Smith'	cell 1,3 [ ]
cell 2,1 3+7i	cell 2,2 [−3.14...3.14]	cell 2,3 [ ]
cell 3,1 [ ]	cell 3,2 [ ]	cell 3,3 5

Cell Array

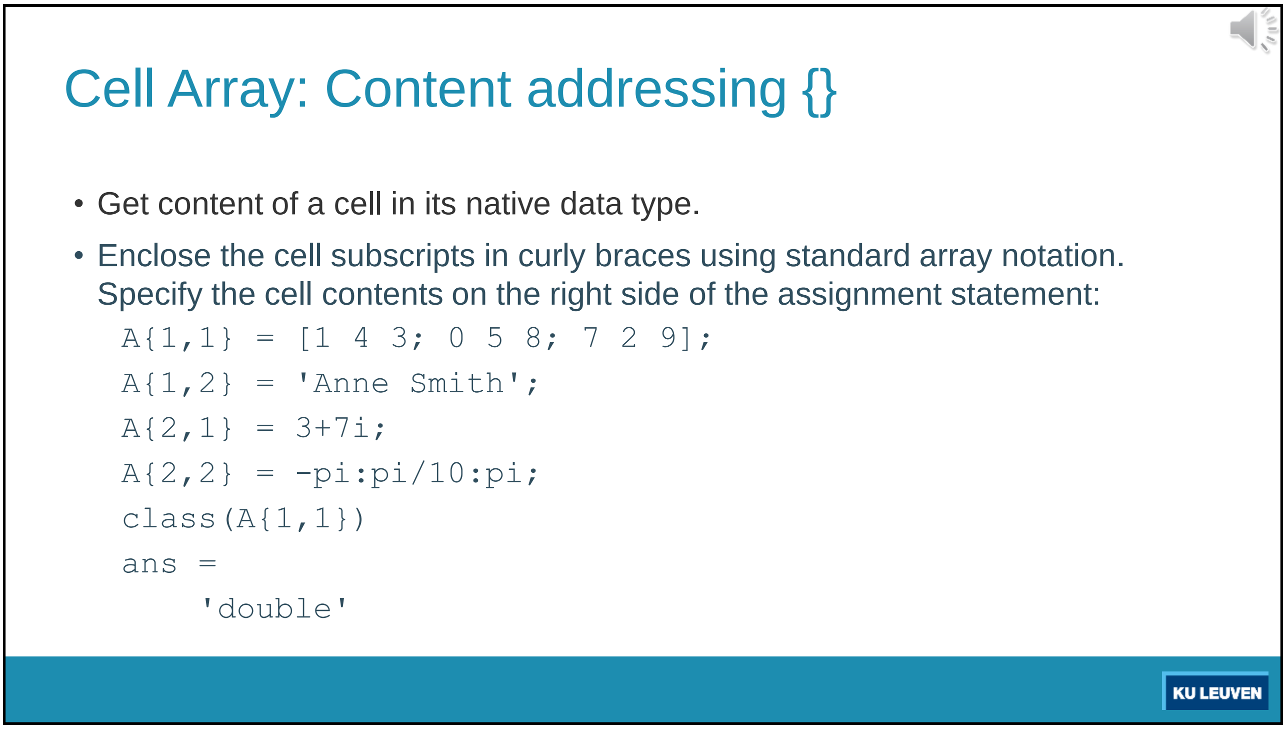
- Cell arrays are created in the same way that data in an array is created and referenced, difference is the use of curly braces { }.
- Cell arrays are used by a lot of built in functions (ie `textscan`, …) and can be particularly useful within scripts.
- Cell arrays should be considered more as data “containers” and must be manipulated accordingly. *(Be careful with the notation when performing arithmetic computations like arrays can, e.g., + - * / ^)*



Cell Array: Cell indexing ()

- Cell indexing allows you to access and manipulate the cells themselves. When accessing the cells themselves, you ignore the content of the cells and merely manipulate the cells.
- Enclose the cell subscripts in parentheses using standard array notation. Enclose the cell contents on the right side of the assignment statement in curly braces {}.

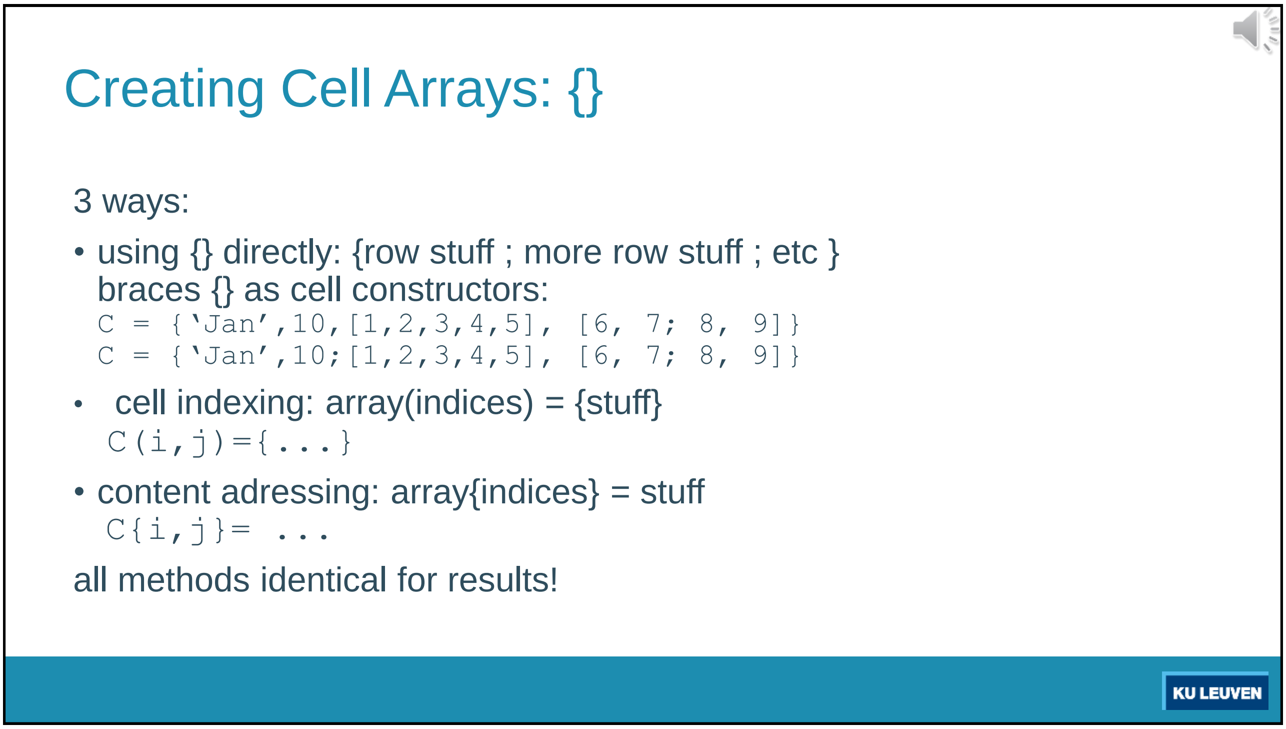
```
A(1,1) = {[1 4 3; 0 5 8; 7 2 9]};
A(1,2) = {'Anne Smith'};
A(2,1) = {3+7i};
A(2,2) = {-pi:pi/10:pi};
class(A(1,1))
ans =
     'cell'
```



Cell Array: Content addressing {}

- Get content of a cell in its native data type.
- Enclose the cell subscripts in curly braces using standard array notation. Specify the cell contents on the right side of the assignment statement:

```
A{1,1} = [1 4 3; 0 5 8; 7 2 9];
A{1,2} = 'Anne Smith';
A{2,1} = 3+7i;
A{2,2} = -pi:pi/10:pi;
class(A{1,1})
ans =
    'double'
```



Creating Cell Arrays: {}

3 ways:

- using {} directly: {row stuff ; more row stuff ; etc }
  braces {} as cell constructors:

```
C = {‘Jan’,10,[1,2,3,4,5], [6, 7; 8, 9]}
C = {‘Jan’,10;[1,2,3,4,5], [6, 7; 8, 9]}
```

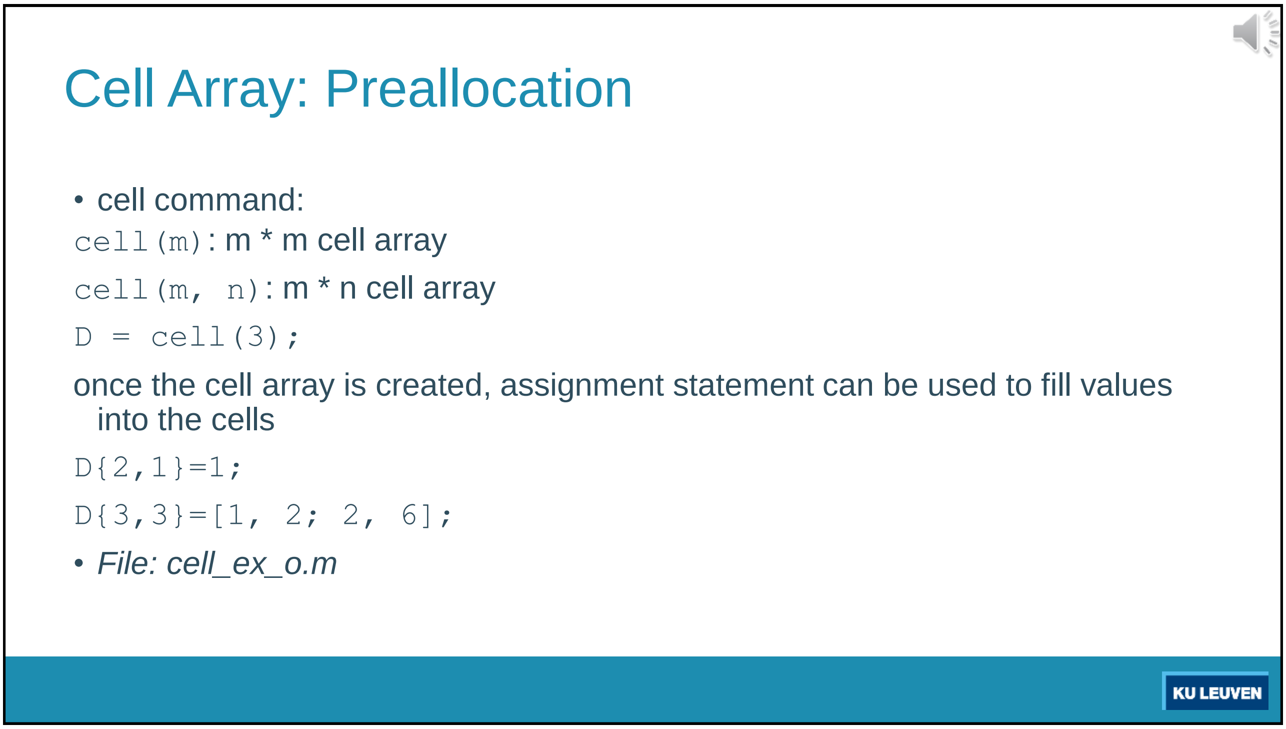
- cell indexing: array(indices) = {stuff}

```
C(i,j)={...}
```

- content adressing: array{indices} = stuff

```
C{i,j}= ...
```

all methods identical for results!



Cell Array: Preallocation

- cell command:

`cell(m)`: m * m cell array

`cell(m, n)`: m * n cell array

```
D = cell(3);
```

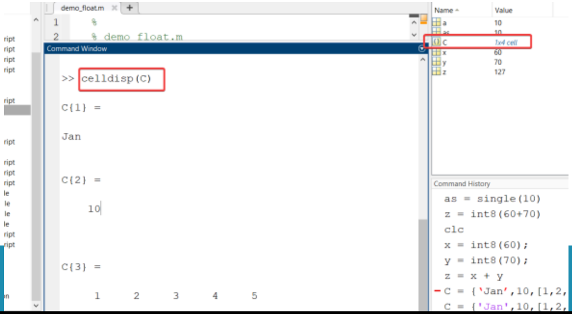
once the cell array is created, assignment statement can be used to fill values into the cells

```
D{2,1}=1;
D{3,3}=[1, 2; 2, 6];
```

- *File: cell_ex_o.m*

Cell: Specific Commands

- `celldisp`: returns the content of a cell array
- `cellplot`: returns graphically the structure of a cell array
- extending a cell array: just by adding
- deleting elements from a cell array: assignment of an empty array
  `D(3,:)=[];`

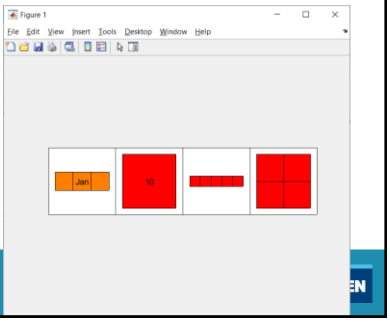


```
1     %
2     % demo_float.m
```

```
>> celldisp(C)

C{1} =

Jan


C{2} =

    10


C{3} =

     1     2     3     4     5
```

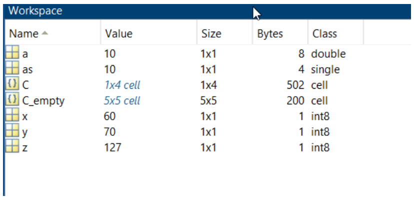
Name	Value
a	10
as	10
C	1x4 cell
x	60
y	70
z	127

```
as = single(10)
z = int8(60+70)
clc
x = int8(60);
y = int8(70);
z = x + y
C = {'Jan',10,[1,2,
C = {'Jan',10,[1,2,
```



Memory requirements

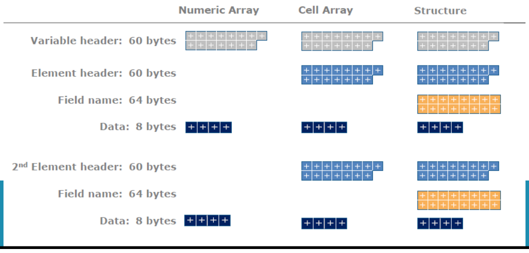
- Cell arrays consume more memory!
- `C = {'Jan',10;[1,2,3,4,5], [6, 7; 8, 9]}`
- `C_empty = cell(5)`
- Check storage requirements for Ce

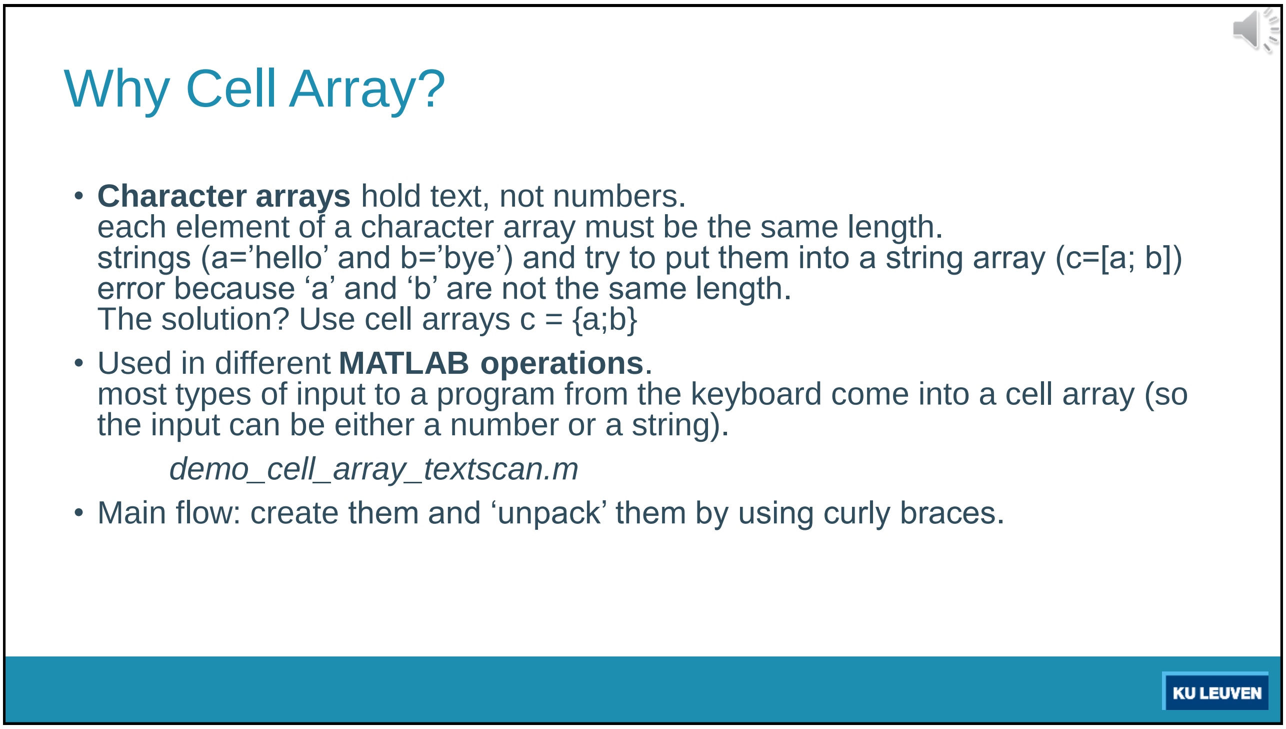


Workspace

Name	Value	Size	Bytes	Class
a	10	1x1	8	double
as	10	1x1	4	single
C	1x4 cell	1x4	502	cell
C_empty	5x5 cell	5x5	200	cell
x	60	1x1	1	int8
y	70	1x1	1	int8
z	127	1x1	1	int8

Container overhead



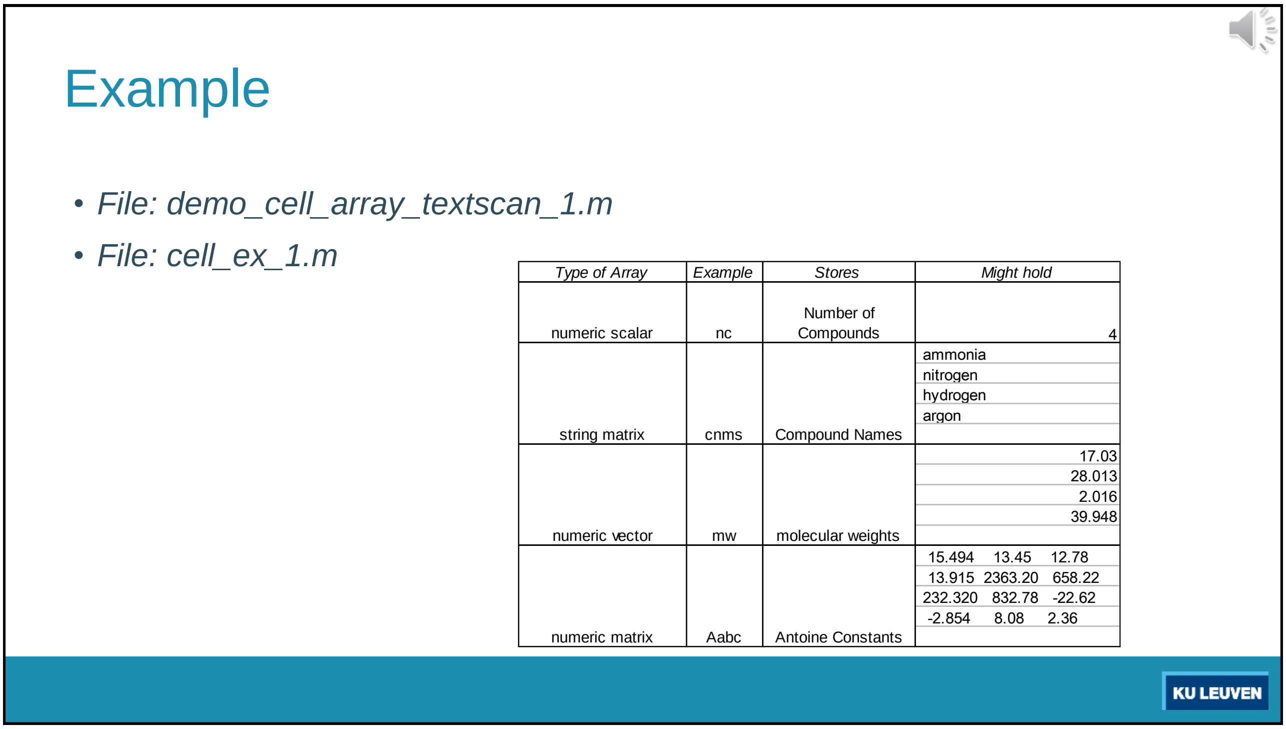


Why Cell Array?

- **Character arrays** hold text, not numbers.
  each element of a character array must be the same length.
  strings (a=’hello’ and b=’bye’) and try to put them into a string array (c=[a; b]) error because ‘a’ and ‘b’ are not the same length.
  The solution? Use cell arrays c = {a;b}
- Used in different **MATLAB operations**.
  most types of input to a program from the keyboard come into a cell array (so the input can be either a number or a string).

  demo_cell_array_textscan.m
- Main flow: create them and ‘unpack’ them by using curly braces.

KU LEUVEN



Example

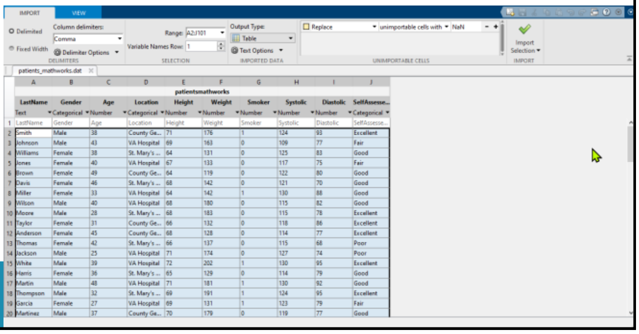
- *File: demo_cell_array_textscan_1.m*
- *File: cell_ex_1.m*

Type of Array	*Example*	*Stores*	*Might hold*
numeric scalar	nc	Number of Compounds	4
string matrix	cnms	Compound Names	ammonia nitrogen hydrogen argon
numeric vector	mw	molecular weights	17.03 28.013 2.016 39.948
numeric matrix	Aabc	Antoine Constants	15.494 13.45 12.78 13.915 2363.20 658.22 232.320 832.78 -22.62 -2.854 8.08 2.36

KU LEUVEN

Example

- import patients_mathworks.dat as cell array
- File: patients_mathworks_cell.mlx



○ Delimited ○ Fixed Width

Column delimiters: Comma

Range: A2:J101

Variable Names Row: 1

Output Type: Table

Replace unimportable cells with NaN

	A	B	C	D	E	F	G	H	I	J
	patientsmathworks									
	LastName	Gender	Age	Location	Height	Weight	Smoker	Systolic	Diastolic	SelfAssesse...
	Text	Categorical	Number	Categorical	Number	Number	Number	Number	Number	Categorical
1	LastName	Gender	Age	Location	Height	Weight	Smoker	Systolic	Diastolic	SelfAssesse...
2	Smith	Male	38	County Ge...	71	176	1	124	93	Excellent
3	Johnson	Male	43	VA Hospital	69	163	0	109	77	Fair
4	Williams	Female	38	St. Mary's ...	64	131	0	125	83	Good
5	Jones	Female	40	VA Hospital	67	133	0	117	75	Fair
6	Brown	Female	49	County Ge...	64	119	0	122	80	Good
7	Davis	Female	46	St. Mary's ...	68	142	0	121	70	Good
8	Miller	Female	33	VA Hospital	64	142	1	130	88	Good
9	Wilson	Male	40	VA Hospital	68	180	0	115	82	Good
10	Moore	Male	28	St. Mary's ...	68	183	0	115	78	Excellent
11	Taylor	Female	31	County Ge...	66	132	0	118	86	Excellent
12	Anderson	Female	45	County Ge...	68	128	0	114	77	Excellent
13	Thomas	Female	42	St. Mary's ...	66	137	0	115	68	Poor
14	Jackson	Male	25	VA Hospital	71	174	0	127	74	Poor
15	White	Male	39	VA Hospital	72	202	1	130	95	Excellent
16	Harris	Female	36	St. Mary's ...	65	129	0	114	79	Good
17	Martin	Male	48	VA Hospital	71	181	1	130	92	Good
18	Thompson	Male	32	St. Mary's ...	69	191	1	124	95	Excellent
19	Garcia	Female	27	VA Hospital	69	131	1	123	79	Fair
20	Martinez	Male	37	County Ge...	70	179	0	119	77	Good

Conversion

- cells and matrices
  - A matrix can be converted to a cell array

```
A=1:4
Acell=num2cell(A)
```

  - elements of a cell can be differently sized matrices, cells can't always be converted to matrices.

```
Amat=cell2mat(Acell)
```

- cells and structs
  - need to specify fieldnames

```
newtpl=cell2struct(tplcell,{'firstname','familyname','height'},2)
```

    The final "2" argument (denoting the 2nd dimension) is necessary, otherwise fieldnames are lost.
  - `tplcell2=struct2cell(tpl)`
- *File: demo_cell_conversion.m*

Have a look at

- http://blogs.mathworks.com/loren/2006/06/21/cell-arrays-and-their-contents/

KU LEUVEN

Structure data type

Class Name	Documentation	Intended Use
`struct`	Structures	• Fields store arrays of varying classes and sizes. • Access one or all fields/indices in single operation. • Field names identify contents. • Method of passing function arguments. • Use in comma-separated lists. • More memory required for overhead

KU LEUVEN

Structure array

- Structures can store different types of data similar to cell arrays, but the data is stored by **name**, `fields` , rather than by index (hierarchy)
- Structures are similar to structures in C

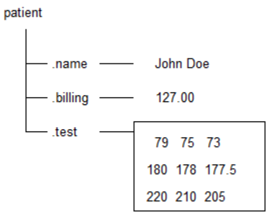
```
A = 1:3;
B = ['abcdefg'];
C = single([1, 2, 3; 4, 5, 6]);
my_struct.numbers = A
my_struct.letters = B
my_struct.singlenumbers = C
```

Structure array

- Can create structure array

```
my_struct(2).numbers = [2 3 6 8 9 10]
```

- *File: strucdem.mlx*



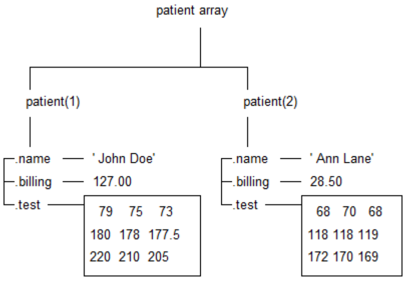
patient

.name — John Doe

.billing — 127.00

.test —

79	75	73
180	178	177.5
220	210	205



patient array

patient(1)

.name — ' John Doe'

.billing — 127.00

.test —

79	75	73
180	178	177.5
220	210	205

patient(2)

.name — ' Ann Lane'

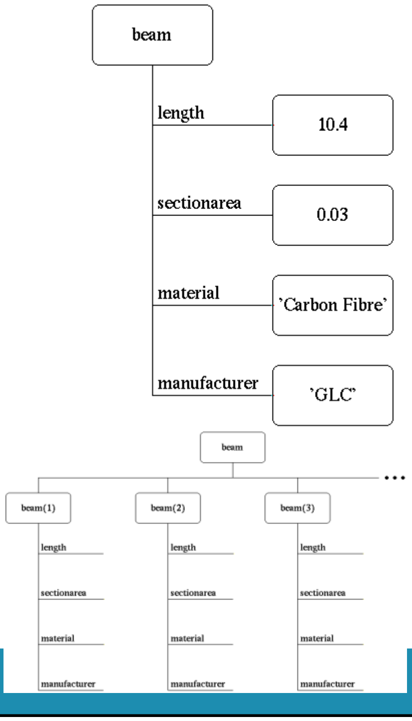
.billing — 28.50

.test —

68	70	68
118	118	119
172	170	169

Structures

- structures are inherently array oriented.
- Contents is addressable by name
- To access these fields, the dot “.” notation is used.
- Each element of a structure can be of a different data type
- Multiple instances of a single structure: build up an array of structures.


beam

length — 10.4

sectionarea — 0.03

material — ’Carbon Fibre’

manufacturer — ’GLC’

beam

beam(1): length, sectionarea, material, manufacturer

beam(2): length, sectionarea, material, manufacturer

beam(3): length, sectionarea, material, manufacturer

…

KU LEUVEN

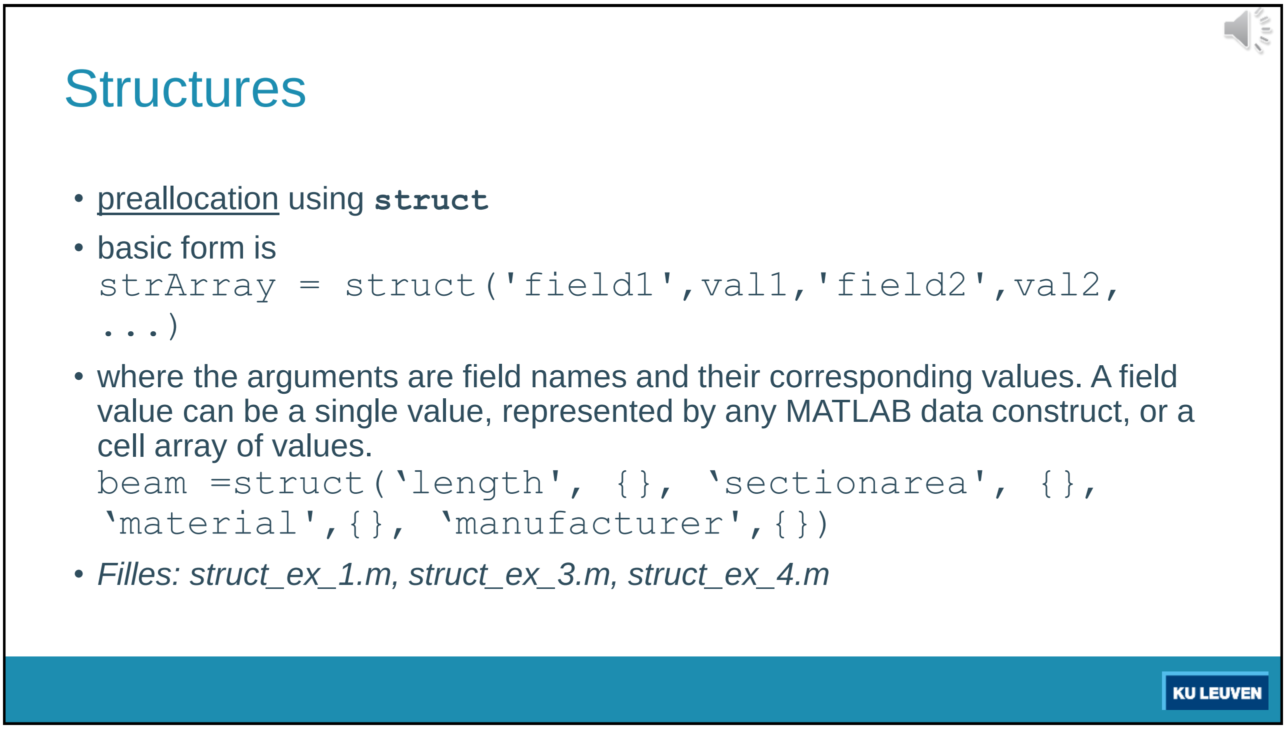
Structures

- Creating structures?
  1. a field at a time by assignment
  2. `struct` function
- Assignment

```
beam.length = 10,4;
beam.sectionarea = 0,03;
beam.material = ‘Carbon Fibre’;
beam.manufacturer = ‘GLC’;
beam(2).length = 15,48;
beam(2).sectionarea = 0,73;
beam(2).material = ‘Steel’;
beam(2).manufacturer = ‘GLC’;
```

- *File: structure_what.m*

KU LEUVEN



Structures

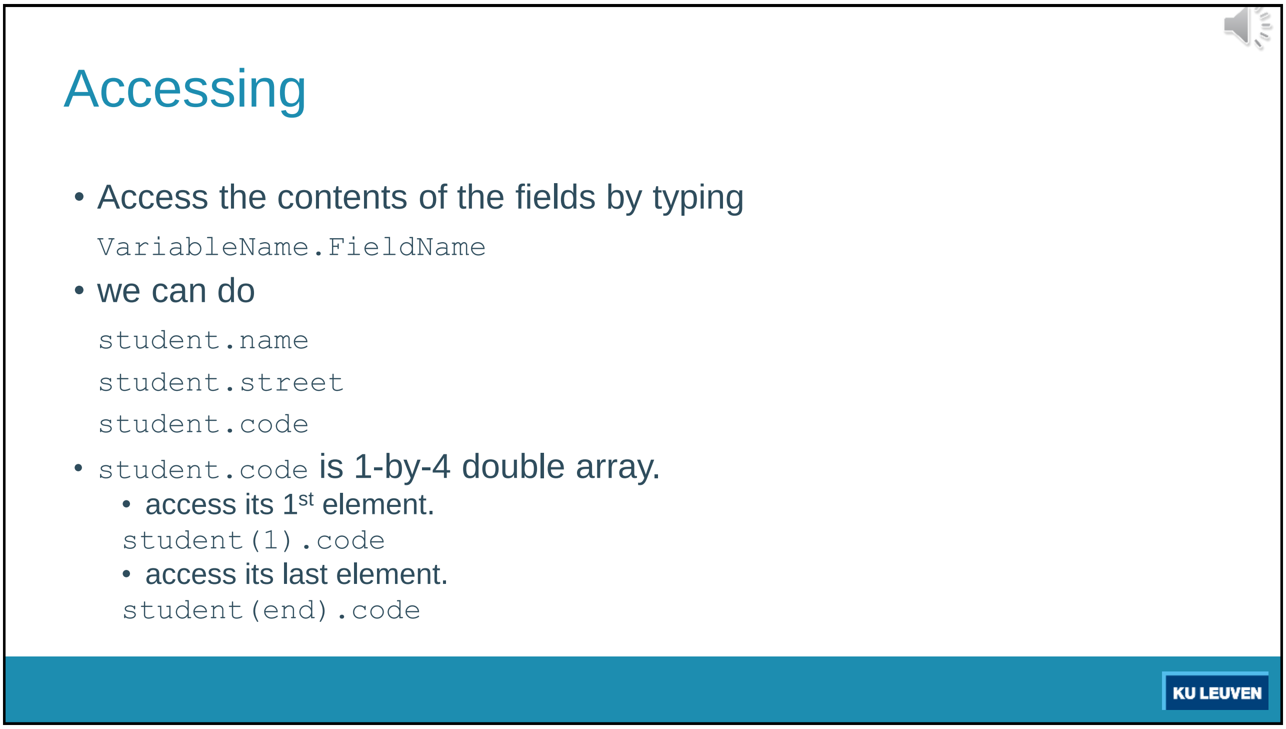
- preallocation using **`struct`**
- basic form is

```
strArray = struct('field1',val1,'field2',val2,
...)
```

- where the arguments are field names and their corresponding values. A field value can be a single value, represented by any MATLAB data construct, or a cell array of values.

```
beam =struct(‘length', {}, ‘sectionarea', {},
‘material',{}, ‘manufacturer',{})
```

- *Filles: struct_ex_1.m, struct_ex_3.m, struct_ex_4.m*



Accessing

- Access the contents of the fields by typing

```
VariableName.FieldName
```

- we can do

```
student.name
student.street
student.code
```

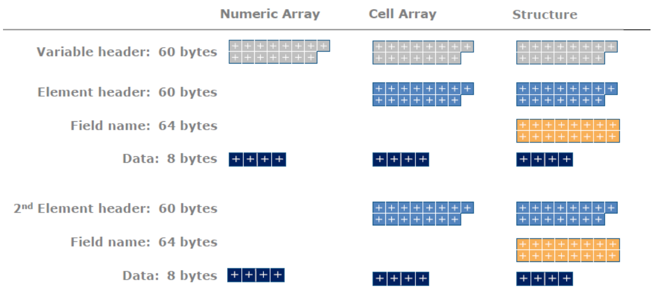
- `student.code` is 1-by-4 double array.
  - access its 1st element.

    ```
    student(1).code
    ```
  - access its last element.

    ```
    student(end).code
    ```

Memory requirements

Container overhead



28

Structures

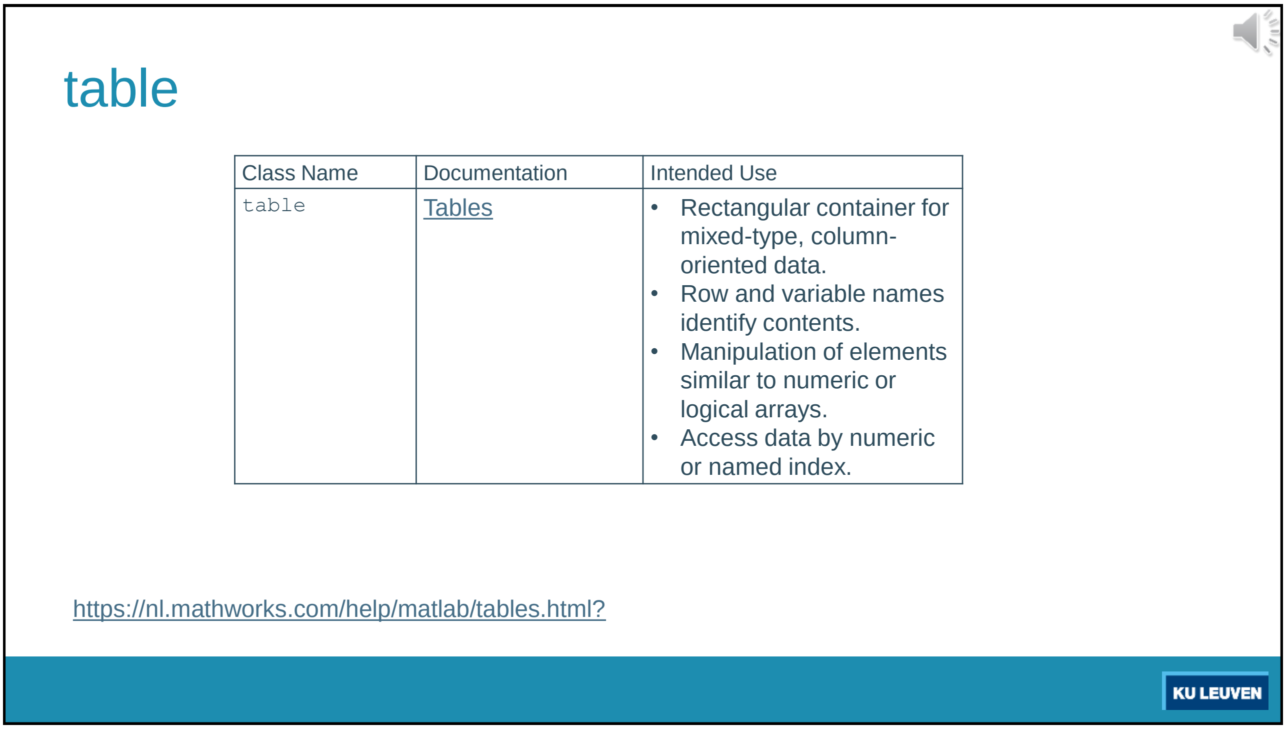
File: demo_structure_textscan_1.m

operations:

- **`rmfield`**: remove a field from a structure
  struct_new = rmfield(struct_old, ‘veld’)
- **`getfield`**: retrieving a value from a field
- **`setfield`**: putting a value in a field
- **`fieldnames`**: returns a list of fieldnames in a cell array of strings

File: struct_ex_2.m

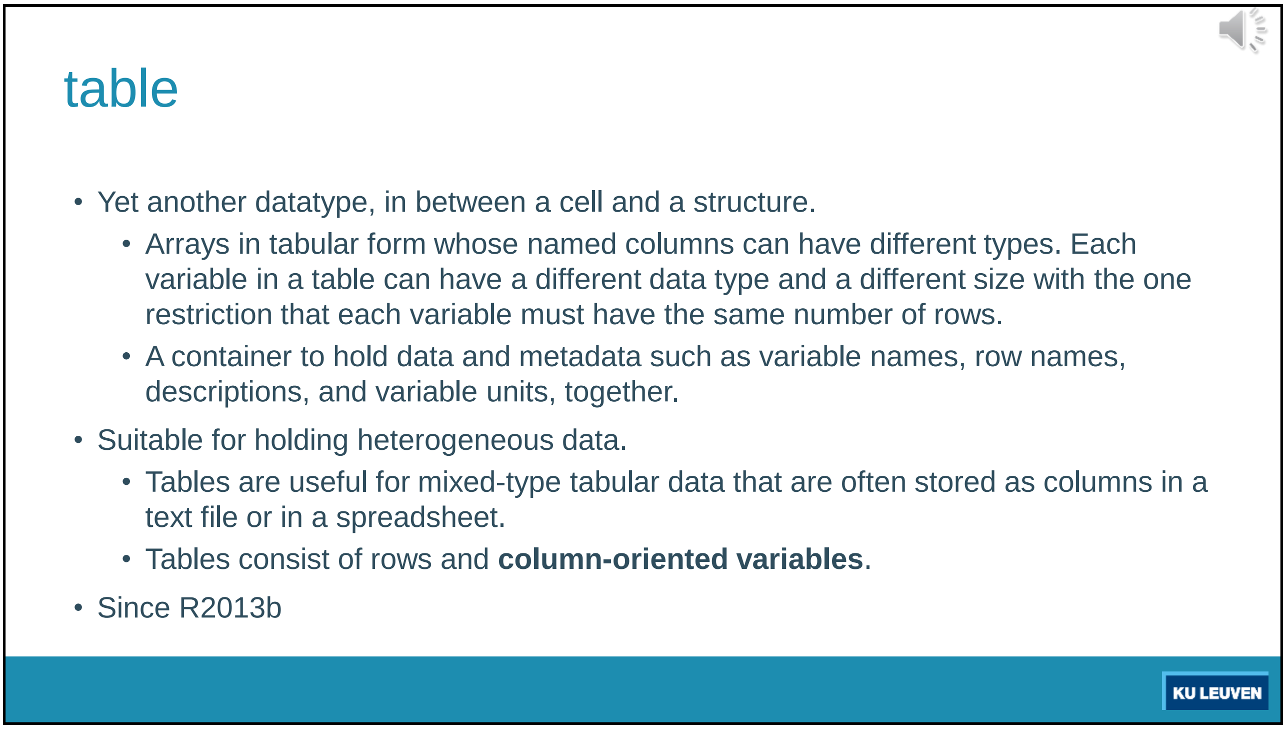
30



table

Class Name	Documentation	Intended Use
`table`	Tables	• Rectangular container for mixed-type, column-oriented data. • Row and variable names identify contents. • Manipulation of elements similar to numeric or logical arrays. • Access data by numeric or named index.

https://nl.mathworks.com/help/matlab/tables.html?



table

- Yet another datatype, in between a cell and a structure.
  - Arrays in tabular form whose named columns can have different types. Each variable in a table can have a different data type and a different size with the one restriction that each variable must have the same number of rows.
  - A container to hold data and metadata such as variable names, row names, descriptions, and variable units, together.
- Suitable for holding heterogeneous data.
  - Tables are useful for mixed-type tabular data that are often stored as columns in a text file or in a spreadsheet.
  - Tables consist of rows and **column-oriented variables**.
- Since R2013b

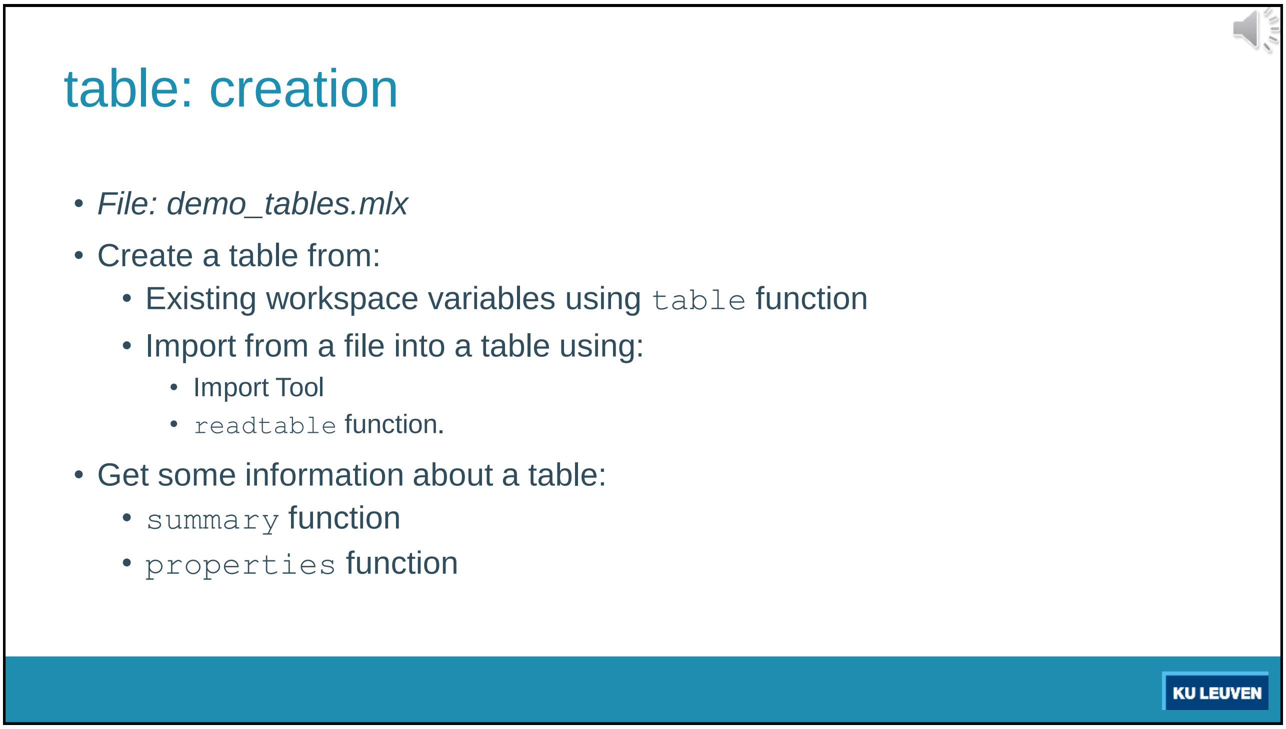


table: creation

- *File: demo_tables.mlx*
- Create a table from:
  - Existing workspace variables using `table` function
  - Import from a file into a table using:
    - Import Tool
    - `readtable` function.
- Get some information about a table:
  - `summary` function
  - `properties` function

KU LEUVEN

34

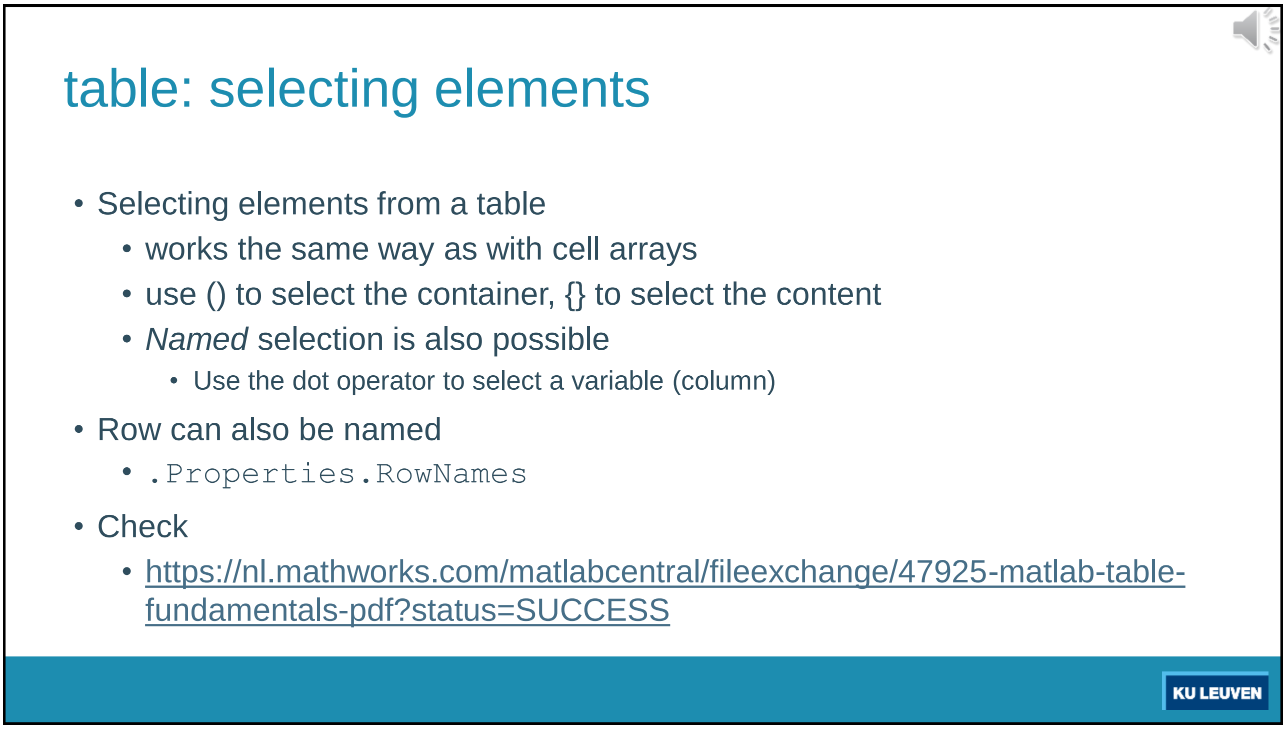


table: selecting elements

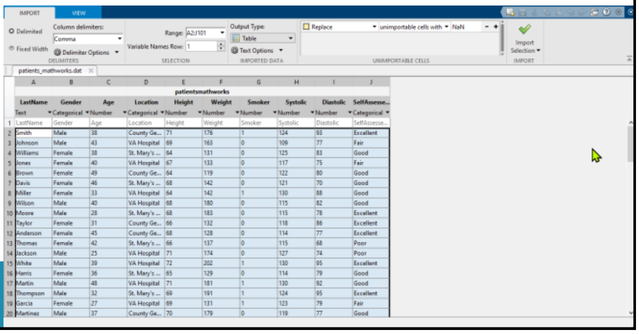
- Selecting elements from a table
  - works the same way as with cell arrays
  - use () to select the container, {} to select the content
  - *Named* selection is also possible
    - Use the dot operator to select a variable (column)
- Row can also be named
  - `.Properties.RowNames`
- Check
  - https://nl.mathworks.com/matlabcentral/fileexchange/47925-matlab-table-fundamentals-pdf?status=SUCCESS

KU LEUVEN

35

Example

- import patients_mathworks.dat as table
- File: patients_mathworks_table.mlx



● Delimited
○ Fixed Width

Column delimiters: Comma

Range: A2:J101

Variable Names Row: 1

Output Type: Table

Replace unimportable cells with NaN

	A	B	C	D	E	F	G	H	I	J
	patientsmathworks									
	LastName	Gender	Age	Location	Height	Weight	Smoker	Systolic	Diastolic	SelfAssesse...
	Text	Categorical	Number	Categorical	Number	Number	Number	Number	Number	Categorical
1	LastName	Gender	Age	Location	Height	Weight	Smoker	Systolic	Diastolic	SelfAssesse...
2	Smith	Male	38	County Ge...	71	176	1	124	93	Excellent
3	Johnson	Male	43	VA Hospital	69	163	0	109	77	Fair
4	Williams	Female	38	St. Mary's ...	64	131	0	125	83	Good
5	Jones	Female	40	VA Hospital	67	133	0	117	75	Fair
6	Brown	Female	49	County Ge...	64	119	0	122	80	Good
7	Davis	Female	46	St. Mary's ...	68	142	0	121	70	Good
8	Miller	Female	33	VA Hospital	64	142	1	130	88	Good
9	Wilson	Male	40	VA Hospital	68	180	0	115	82	Good
10	Moore	Male	28	St. Mary's ...	68	183	0	115	78	Excellent
11	Taylor	Female	31	County Ge...	66	132	0	118	86	Excellent
12	Anderson	Female	45	County Ge...	68	128	0	114	77	Excellent
13	Thomas	Female	42	St. Mary's ...	66	137	0	115	68	Poor
14	Jackson	Male	25	VA Hospital	71	174	0	127	74	Poor
15	White	Male	39	VA Hospital	72	202	1	130	95	Excellent
16	Harris	Female	36	St. Mary's ...	65	129	0	114	79	Good
17	Martin	Male	48	VA Hospital	71	181	1	130	92	Good
18	Thompson	Male	32	St. Mary's ...	69	191	1	124	95	Excellent
19	Garcia	Female	27	VA Hospital	69	131	1	123	79	Fair
20	Martinez	Male	37	County Ge...	70	179	0	119	77	Good