Matlab Tutorials

Das *LiveScript* zu dem Tutorial 4.

Lösung: Aufgabe aus Tutorial 3

```
test = zeros(7)
test(2:end-1, 2:end-1) = 1
test(3:end-2, 3:end-2) = 2
test(round(end/2), round(end/2)) = 3
```

Tutorial 4.1

- Datentypen in Matlab (eine Auswahl)

Alle von Matlab unterstützten Datentypen findet man mit *doc Language Fundamentals* unter "Data Types".

Datentyp **integer / double**

```
iNum1 = 7;           % double, Standard in Matlab
iNum2 = int32(7);    % 32-bit integer
iNum3 = ones(3, 1);
iNum4 = ones(3, 5);
dNum5 = sqrt(2);     % in Command Line werden nur 4 Dezimalstellen angezeigt
```

Datentyp **bool**

```
bBedingung = 2+1 == 3;
```

Datentyp **string**

```
szTest1 = 'Dies ist ein String!';
szTest1(6:8)           % Teil des Strings
% ersetzen von Zeichen, hier: i => I
szTest1_neu = strrep(szTest1, 'i', 'I');
strcmp('Signal', 'signal')
strcmpi('Signal', 'signal')
```

Strings können auch Zeilenweise geschrieben werden. **Achtung:** Alle Strings müssen dabei die selbe Länge haben!

```
szTest2 = ['test1'; 'test2'; 'test3'; 'test4'];    % Liste von Strings
szTest3 = ['test11'; 'test2'; 'test3'; 'test4'];   % FEHLER: Strings haben
                                                   % unterschiedliche Längen
                                                   % => cell arrays
```

Zusammenfügen von strings:

```
iNum = 5;
szTest4 = ['Test-Nummer: ' num2str(iNum)];
szTest5 = sprintf('Test-Nummer: %i', iNum);
szTest6 = sprintf('%8.2f', sqrt(2));
length(szTest6)
szTest7 = sprintf('Test-Nummer: %8.2f', sqrt(2));
```

Datentyp **cell array**

```
caTest1 = {'test1', 'test11', 3, sqrt(2), ones(3, 3)};
szInhalt = caTest1{1};    % Inhalt der ersten Zelle
caZelle  = caTest1(1);    % gesamte erste Zelle
caTest1{1}(1:end-1)       % Indizierung auch in einer Zelle möglich

caTest2 = {'TEST', zeros(4), sqrt(2), caTest1};
caTest2{end}              % entspricht caTest1 (1x5 cell)
caTest2{end}{3} == caTest1{3}
caTest2(end)              % Zelle (1x1 cell), die anderes cell array enthält
```

Datentyp **struct**

```
stData1 = struct('Feld1', {1, 2}, 'Feld2', {11, 22});
stData1(3).Feld1 = 3;
stData1(3).Feld2 = 33;
```

Zugriff auf *struct*:

```
stData1(2)
[stData1.Feld1]
{stData1.Feld1}

szFieldname = 'Feld1';
stData1.(szFieldname)

fieldnames(stData1)
```

Datentyp **handles**

```
hFigure = figure('Name', 'Testgrafik');
hAxes = gca;
hFunction = @ones;
hAnoFunction = @(x) (x + 1)^2;
```

Datentyp **dates and time**

```
tNow         = datetime();
iLastYear    = tNow.Year - 1;      % double: letzte Jahr (z.B. 2015)
```

```
t1YearBefore = tNow - years(1);    % Zeitpunkt: letztes Jahr (z.B. 15-Jul-2016 09:15:00)
tSeries      = tNow + hours(1:3);  % mehrere Zeitpunkte

tNow.Format = 'dd-MM-yy'     % Ausgabe verändern

% aus Doc
caDateStrings = {'2014-05-26';'2014-08-03'};
t = datetime(caDateStrings,'InputFormat','yyyy-MM-dd')

% mehr Informationen unter
doc Date and Time Arithmetic
```

Tutorial 4.2

- Funktionen

So wie Skripte nur mit Syntax:

```
function [out1, out2] = meineFunktion(in1, in2)
% MEINEFUNKTION wird tolle Dinge tun!
%
% MEINEFUNKTION contains some very usefull calculations from the
% "Matlab Tutorials".

% Author: Julian Kahnert (c) IHA @ Jade Hochschule


    out1 = in1 + in2;
    out2 = in1 * in2;

end
```

Achtung: Jede Funktion muss sich in einer eigenen Datei befinden, welche dem Funktionsnamen entspricht (hier also: "meineFunktion.m").

Für einen variablen Input/Output sollte man sich die Dokumentation der Befehle *varargin* und *varargout* genauer ansehen.

Tutorial 4.3

- Schleifen: *for*
- Schleifen: *while*

for-Schleife: Anzahl der Schleifendurchläufe ist zu Beginn bekannt.

```
for n = 1:5
    disp(n)
end
```

```
vTests = [2, 12, 13, 7];
for nn = vTests
    disp(nn)
```

```
end
```

```
caSatz = {'Das' 'hier' 'ist' 'ein' 'Test!'};
for k = 1:length(caSatz)
    szAktuellerString = caSatz{k};
    disp(szAktuellerString)
end
```

while-Schleife: Anzahl der Schleifendurchläufe ist zu Beginn *nicht* bekannt.

```
kk = 0
while kk < 10
    disp(kk)
    kk = kk + 1;
end
```

Nativ keine fußgesteuerten Schleifen in Matlab, Abhilfe: while-Schleife mit *break.*

Tutorial 4.4

- Abbruchkriterium: *continue*
- Abbruchkriterium: *break*
- Abbruchkriterium: *return*

```
for n = 1:5
    if n == 3
        continue     % überspringt diesen Schleifendurchlauf
    end

    disp(n)
end
disp(['Letzter Index:' num2str(n)])
```

```
for n = 1:5
    if n == 3
        break       % beendet Schleife
    end

    disp(n)
end

% wird noch ausgeführt
disp(['Letzter Index:' num2str(n)])
```

```
for n = 1:5
    if n == 3
        return       % beendet Funktion/Skript
```

```
    end

    disp(n)
end

% wird nicht mehr ausgeführt
disp(['Letzter Index:' num2str(n)])
```

Achtung: Der Befehl *return* beendet nicht nur die Schleife sondern die komplette Funktion!

Implementierung einer fußgesteuerten Schleifen in Matlab:

```
n = 0;
while true
    disp(n)

    n = n + 1;
    if n > 5
        break
    end
end
```

Tutorial 4: Aufgabe

Skript vervollständigen und Funktion (*tut4_function.m*) schreiben, die alle Input Variablen in einem *struct* zusammenfügt und in einer Output Variable abspeichert.

```
iIn  =      % integer/double
bIn  =      % bool
szIn =      % string
caIn =      % cell array
hIn  =      % handle
tIn  =      % datetime object

stOut = tut4_function(iIn, bIn, szIn, caIn, hIn, tIn);

disp(stOut)
```