

Tutorial – Handout zur Lerndokumentation

MATLAB → nutzerdefinierte Funktionen

- **Vorteile von nutzerdefinierten Funktionen**
 - Wiederverwendbar
 - Übersicht
 - Notwendige Parameter erkennbar
 - Strukturierung des Hauptprogramms
 - Gemeinsame Bearbeitung möglich
- **Speicherung der eigenen Funktionen**
 - am Ende des Hauptprogramms
 - als separate .m-Datei abgespeichert → Datei muss identischen Namen wie die Funktion haben und im selben Pfad liegen

- **Syntax**

```
function [rueckgabewert_1, rueckgabewert_2] = name_funktion(x, y, z)
    lokal = sqrt(4 * x / y);

    rueckgabewert_1 = (x + z) * lokal;
    rueckgabewert_2 = (y + z) * lokal;
end
```

- Abspeicherung als name_funktion.m
 - rueckgabewert_1 / rueckgabewert_2: Name ist frei wählbar, Ergebnis(se) der Funktion
 - x, y, z: Inputargumente → Funktion erhält Kopie der Werte aus Hauptprogramm
 - lokal: Variable nur innerhalb der Funktion sichtbar
- **Rückgabewerte**
 - Einzelne Variable

```
function [rueckgabewert] = name_funktion(x, y, z)...
```

- Mehrere Variablen als Rückgabewerte in Klammern

```
function [rueckgabewert_1, rueckgabewert_2] = name_funktion(x, y, z)...
```

- Rückgabe optional

```
function name_funktion(x, y, z)...
```

- **Funktion im Hauptprogramm abrufen**

```
% Aufruf der Funktion name_funktion im Hauptprogramm
[wert_1, wert_2] = name_funktion(a, b, c)
```

MATLAB → Das Struct

- **Datenstruktur**

- eine Hauptvariable zur Ansteuerung
- Untervariablen zur Speicherung von Werten, Arrays, Matrizen, etc...

- **Vorteile eines Structs**

- Strukturiertes Speichern von Variablen
- Gruppieren von inhaltlich zusammenhängenden Variablen
- Schneller Zugriff auf Variablen

- **Definition**

- Mehrere Werte bei der Erzeugung übergeben

```
name_struct = struct('field1',value1,'field2',value2,...,'fieldN',valueN)

% Bsp.:
Beispiel = struct('x',3, 'y', 7)
```

- Einzelne Werte übergeben

```
name_struct.subvariable = value

% Bsp.:
Beispiel.x = 3
Beispiel.y = 7
```

- **Auslesen**

- Alle Werte/Untervariablen gleichzeitig auslesen

```
name_struct
```

- Struct vollständig in anderer Variablen Speichern

```
a = name_struct
```

- Einzelne Werte/Untervariablen auslesen

```
name_struct.subvariable
```

- Einzelne Werte/Untervariablen übergeben

```
a = name_struct.subvariable

% Bsp.:
a = Beispiel.x
```