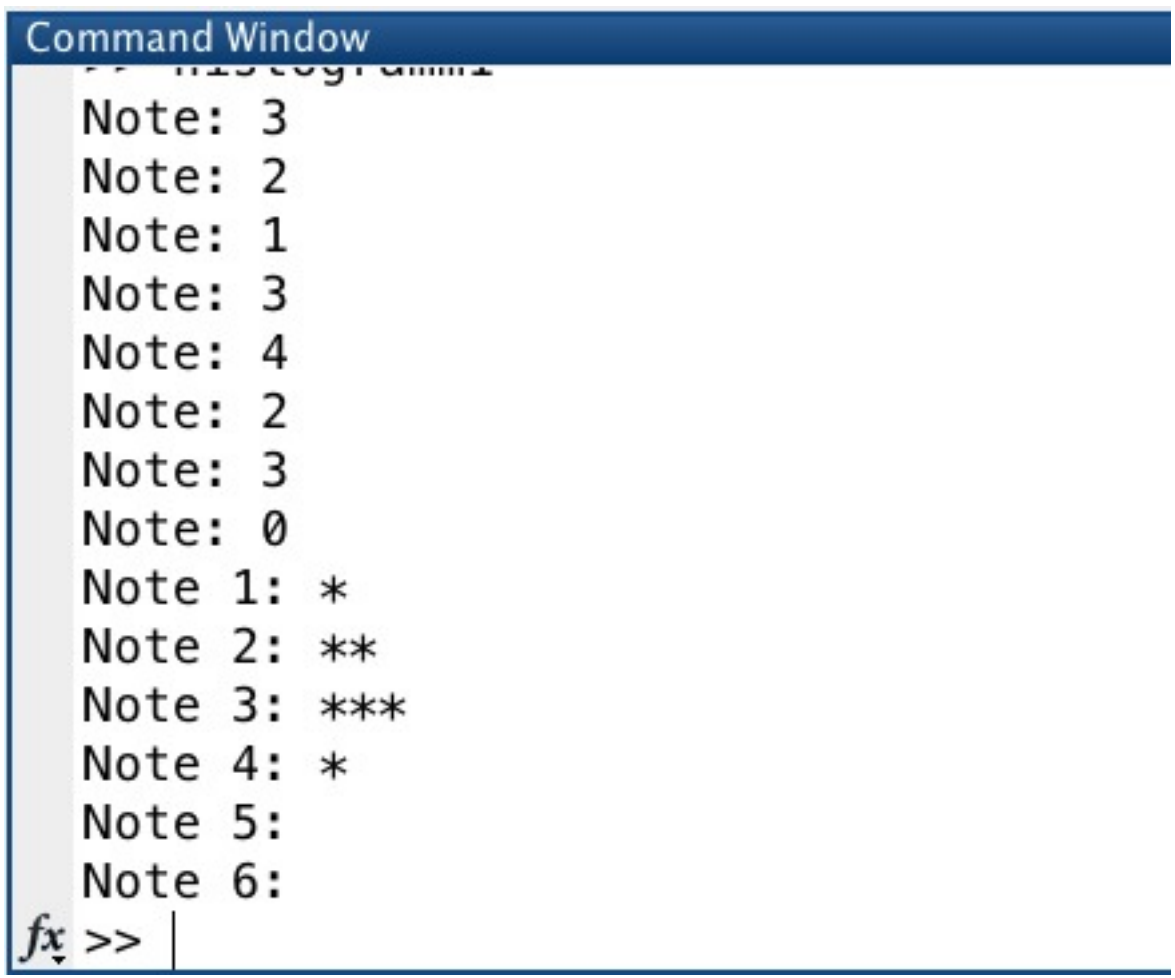
Grundlagen der Programmierung

11. Vorlesung
09.01.2018

Arrays, Vektoren, Matrizen

Beispiel: Notenhistogramm

- Aufgabe: Zahlen (Noten von 1-6) von Tastatur einlesen, bis 0 eingegeben wird, Histogramm ausgeben



n Sterne ausgeben

```
function print_n_stars( n )
% This prints n stars
    for x = 1:n
        fprintf('*');
    end
end
```

Notenhistogramm: Eingabe

```
n1 = 0; n2 = 0; n3 = 0; n4 = 0; n5 = 0; n6 = 0;
eingabe = 1;
while eingabe ~= 0
    eingabe = input('Note: ');
    switch eingabe
        case 1
            n1 = n1 + 1;
        case 2
            n2 = n2 + 1;
        case 3
            n3 = n3 + 1;
        case 4
            n4 = n4 + 1;
        case 5
            n5 = n5 + 1;
        case 6
            n6 = n6 + 1;
    end
end
```

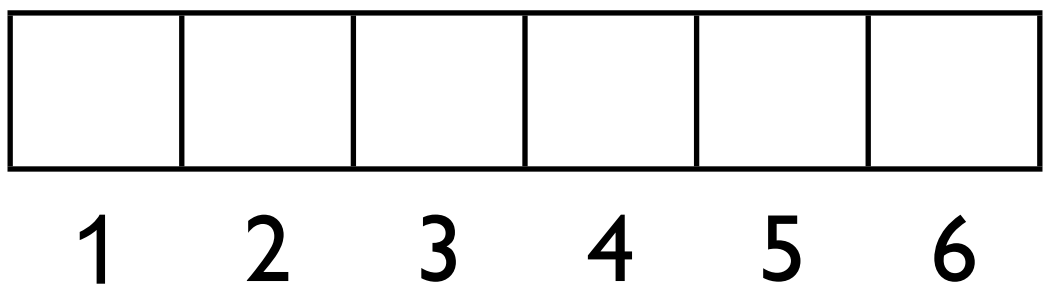
Notenhistogramm: Ausgabe

```
fprintf('\nNote 1: ');
print_n_stars( n1 );
fprintf('\nNote 2: ');
print_n_stars( n2 );
fprintf('\nNote 3: ');
print_n_stars( n3 );
fprintf('\nNote 4: ');
print_n_stars( n4 );
fprintf('\nNote 5: ');
print_n_stars( n5 );
fprintf('\nNote 6: ');
print_n_stars( n6 );
```

Geht das einfacher/ übersichtlicher?

Arrays (Felder, Vektoren)

- Container für mehrere Objekte
- Objekte in dem Container werden über *Indizes* adressiert (beginnend bei 1)



- Für Indizes dürfen Ausdrücke benutzt werden (d.h. Indizes können berechnet werden)
- In Matlab: alle Variablen sind Felder

Felder: erzeugen

- Zeilenvektoren:
  `a = [1 2 3 4]`
- Spaltenvektoren:
  `b = [5; 6; 7; 8]`
- Matrizen:
  `c = [1 2; 3 4]`
- Bereiche:
  - `d = 1:5`
  - `e = 1:2:10`

```
a =

     1     2     3     4

b =

     5
     6
     7
     8

c =

     1     2
     3     4

d =

     1     2     3     4     5

e =

     1     3     5     7     9
```

Felder: Zugriff auf Elemente

- Zeilen/Spaltenindex: (Name der Matrix sei `x`)
  `x(1,2)`
- Bereiche (z.B. erste Zeile, Spalten 1 und 3):
  `x(1, [1 3])`
- Bereich (: steht für 'alle', `end` für 'letzter Index')
  `x(1, :) x(1, [2:end])`
- Elemente ändern: alle Arten der Indizierung dürfen auch auf der linken Seite einer Zuweisung stehen
  `x(1,2) = 10`

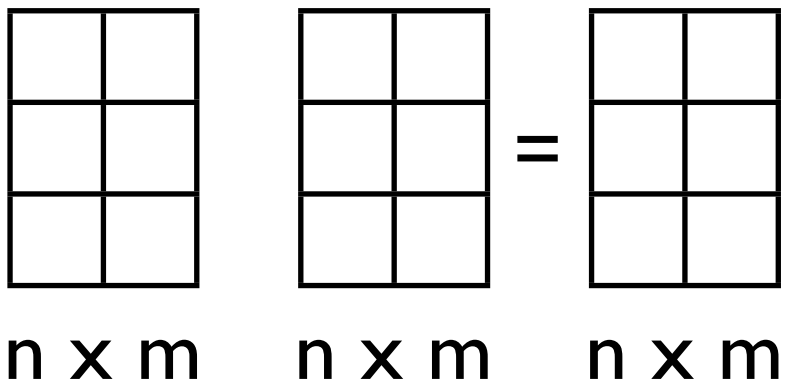
Felder: Elemente hinzufügen/löschen

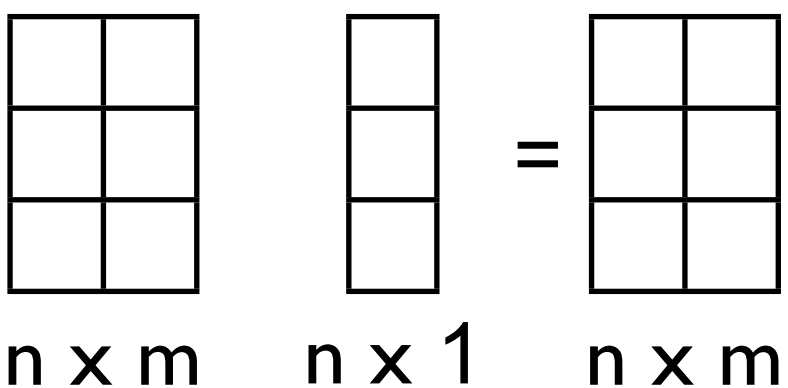
- Hinzufügen:
  - Element mit neuem Index zuweisen
  - Matrix wird evtl. mit Nullen aufgefüllt
- Löschen:
  - Elemente auf "leere Matrix" (`[]`) setzen
  - Z.B.: zweite Spalte löschen:
    ```
    x(:,2) = []
    ```

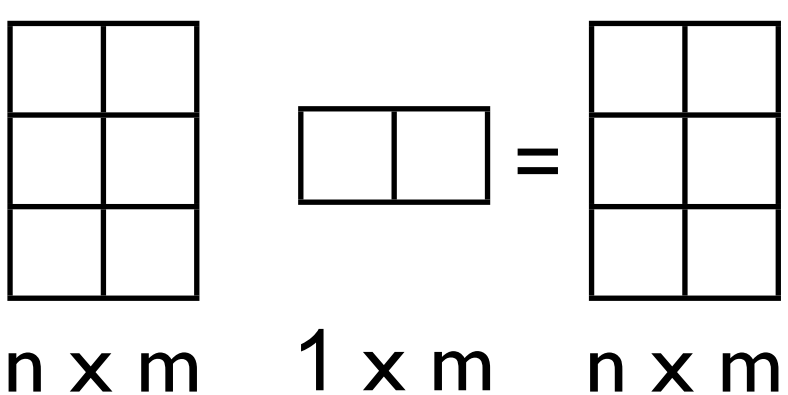
Felder: Operationen

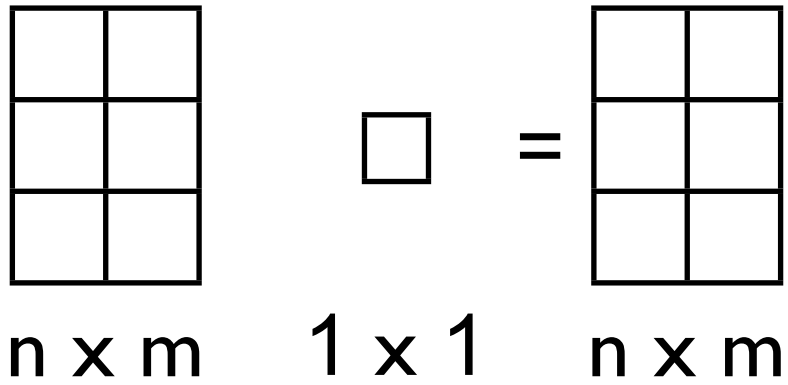
- Matlab unterscheidet:
  - Array-Operationen – `'  .+  .-  .*  ./  .^`
    - elementweise
    - Felder müssen bgzl. Größe ‘kompatibel’ sein
    - beginnen mit . (z.B. `.*`)
    - auch auf mehrdimensionale Felder anwendbar
  - Matrix-Operationen
    - aus linearer Algebra

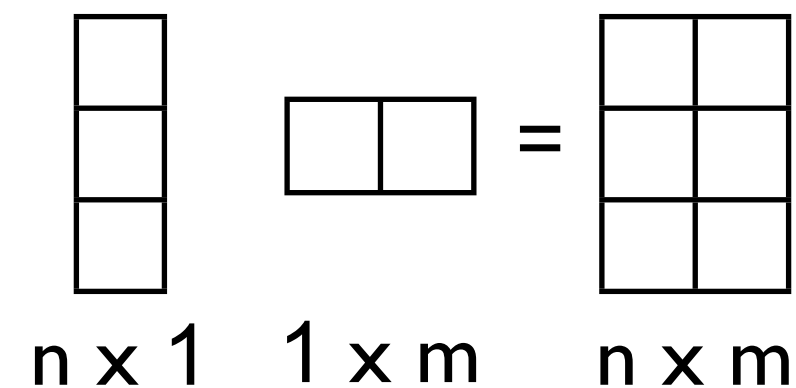
Array Op.: Kompatible Größen











Z.B.: Multiplikationstabelle:

```
[1:10]' .* [1:10]
```

Matrix Operationen

- `C=A*B` Matrixmultiplikation $c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$
- `x=A\B` Lösung des Gleichungsystems *A x* = *B*
- `x=B/A` Lösung des Gleichungsystems *x A* = *B*
- `C=A^B` Potenzierung (A oder B skalar)

Beispiel: Max. Element in Matrix

```
zz = randi(100, 1, 100);
fprintf('max. Element ist: %d\n',  maxi(zz));
fprintf('(mit Matlab Fkt: %d)\n',  max(zz));

function m = maxi( v )
    max_so_far = 0;
    for x = v
        if max_so_far < x
            max_so_far = x;
        end
    end
    m = max_so_far;
end
```

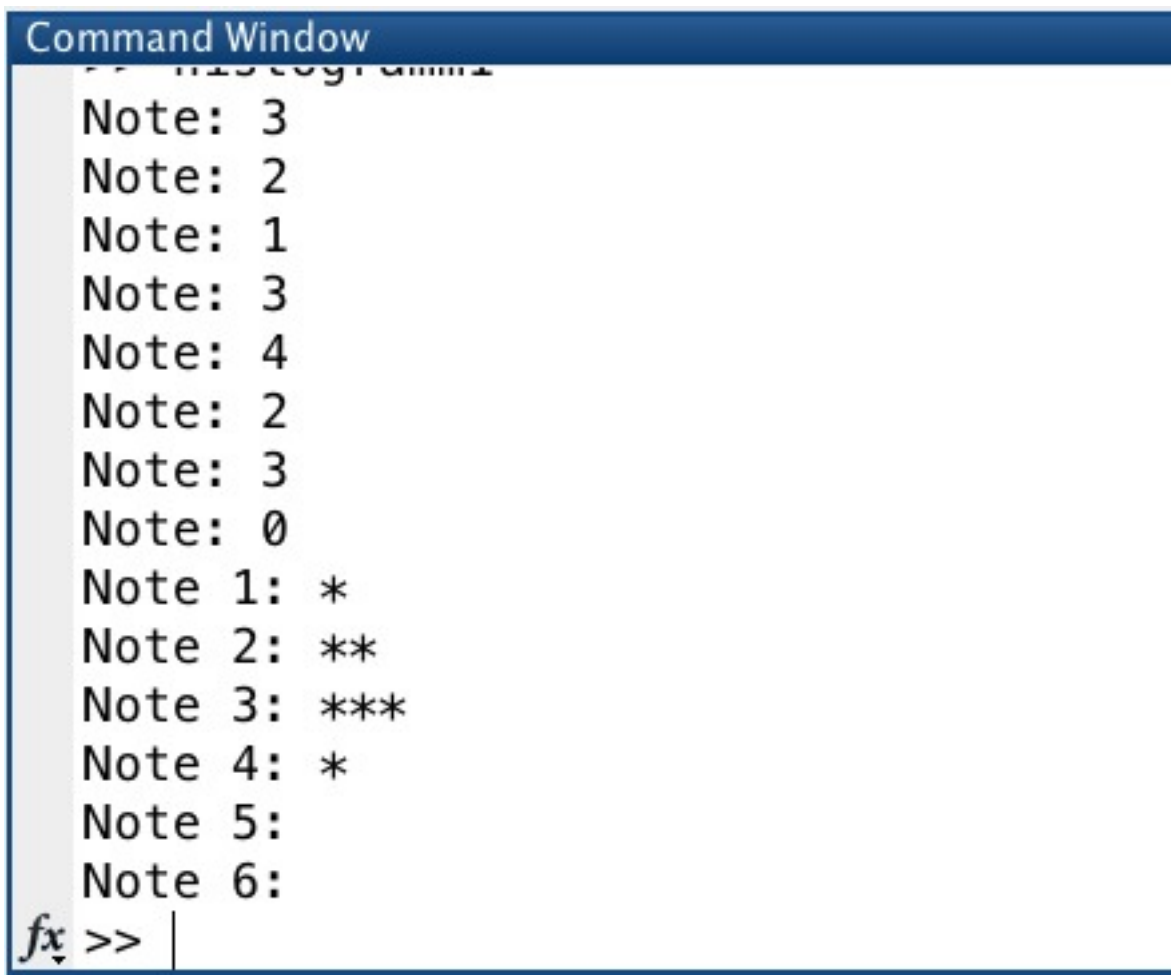
```
>> max_script
max. Element ist: 67
(mit Matlab Fkt: 67)
>>
```

Beispiel: Max. Element mit Index

```
zz = randi(100, 1, 10);
[e,x] = maxi(zz));
fprintf('max. Element ist: %d an %d\n',e,x);

function [m, max_idx] = maxi( v )
   max_so_far = v(1);
   for idx = [2:length(v)]
       if max_so_far < v(idx)
           max_so_far = v(idx);
           max_idx = idx;
       end
   end
   m = max_so_far;
end
```

Zurück zum Notenhistogramm

- Aufgabe: Zahlen (Noten von 1-6) von Tastatur einlesen, bis 0 eingegeben wird, Histogramm ausgeben



Notenhistogramm (neu)

```
n = zeros(1,6);
eingabe = 1;
while eingabe ~= 0
    eingabe = input('Note: ');
    if (eingabe > 0) && (eingabe < 7)
        n(eingabe) = n(eingabe) + 1;
    end
end

for x=[1:6]
   fprintf('\nNote %d: ', x);
   print_n_stars( n(x) );
end
```

Vektor erzeugen

Element erhöhen

Element ausgeben

Notenhistogramm (Version 3)

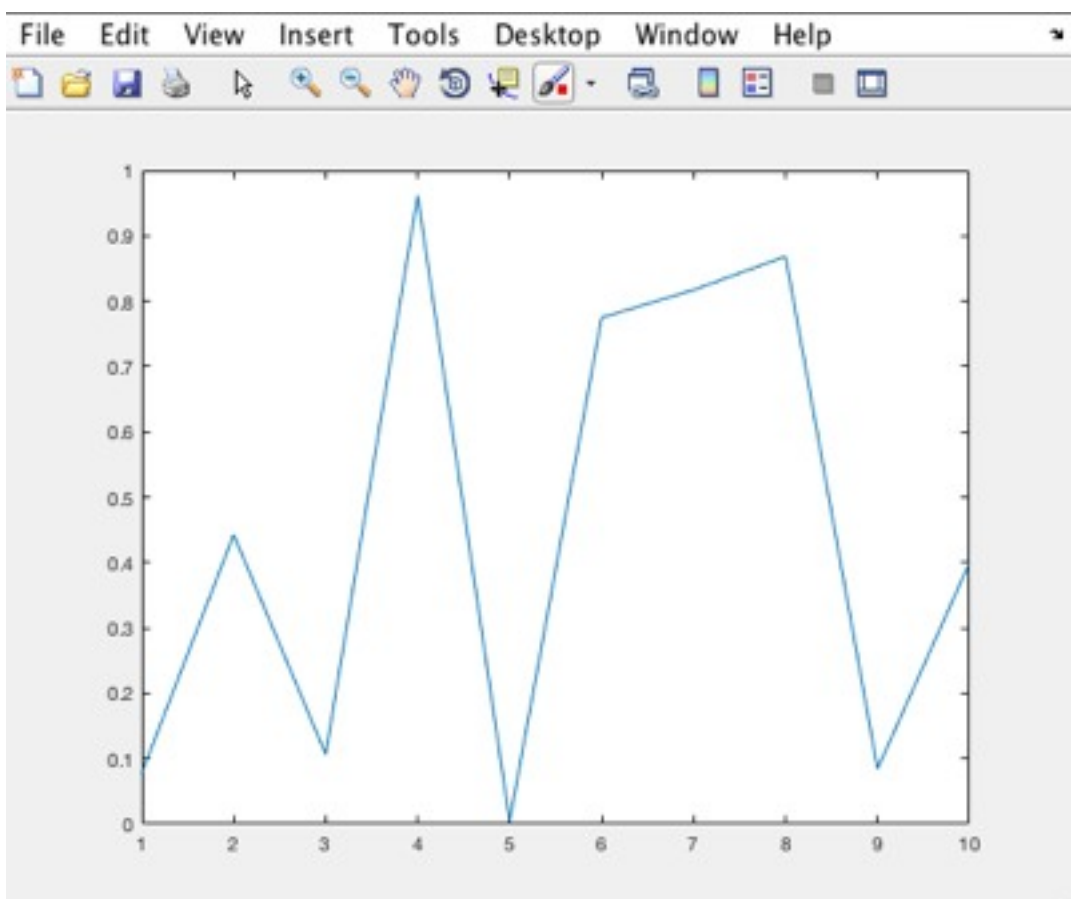
```
spektrum = 6;
n = zeros(1, spektrum);
eingabe = 1;
while eingabe ~= 0
    eingabe = input('Note: ');
    if (eingabe > 0) && (eingabe <= spektrum)
       n(eingabe) = n(eingabe) + 1;
    end
end

for x = [1:spektrum]
   fprintf('\nNote %d: ', x);
   print_n_stars( n(x) );
end
```

Leichter zu Ändern!

```
Note 1: *
Note 2: **
Note 3: ***
Note 4: *
Note 5:
Note 6:
```

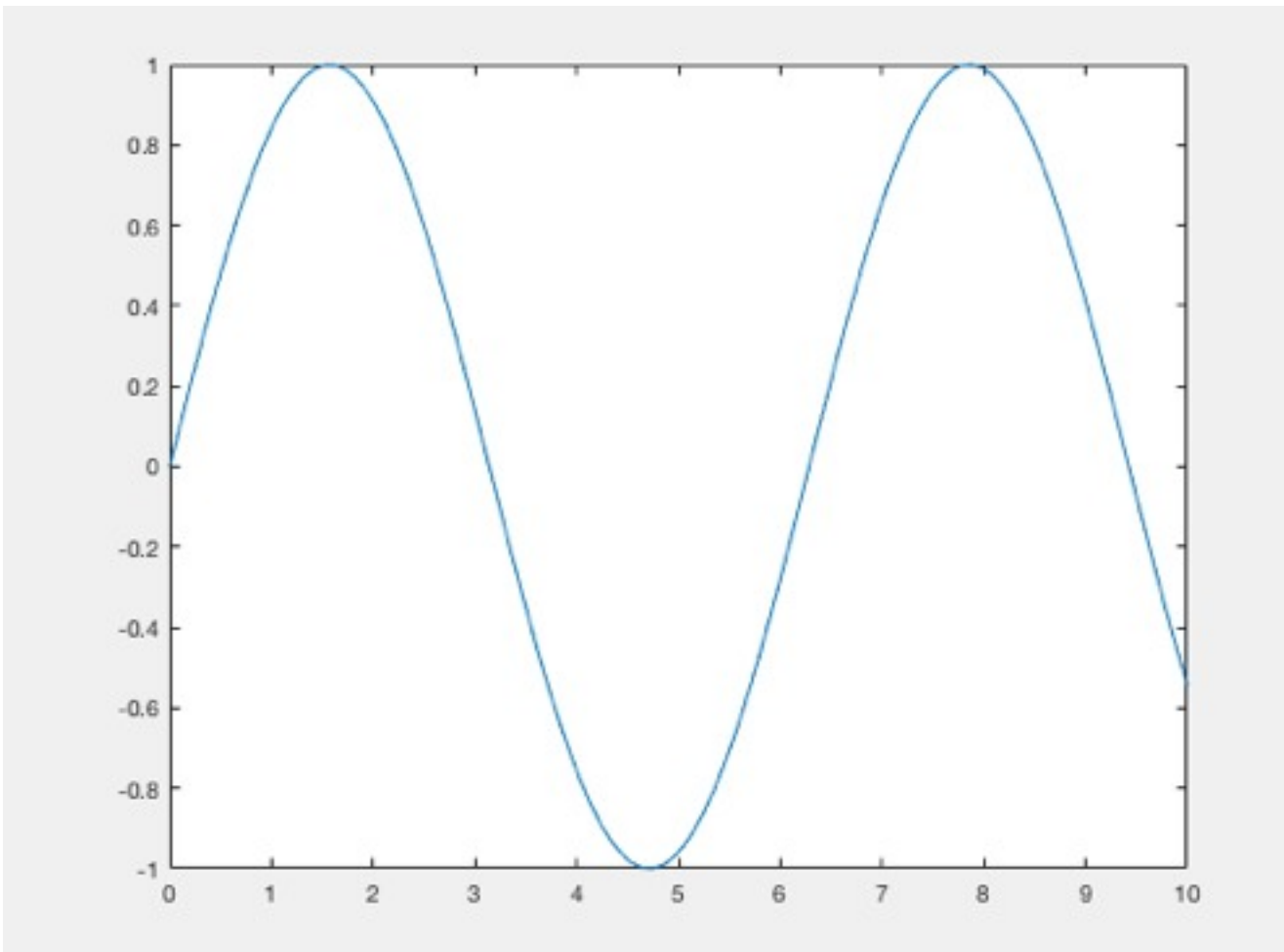
Exkurs: Plots

```
>> x=[1:10]

x =

     1     2     3     4     5     6     7     8     9    10

>> y=rand(1,10)

y =

    0.0782    0.4427    0.1067    0.9619    0.0046    0.7749    0.8173    0.8687    0.0844    0.3998

>> plot(x,y)
```

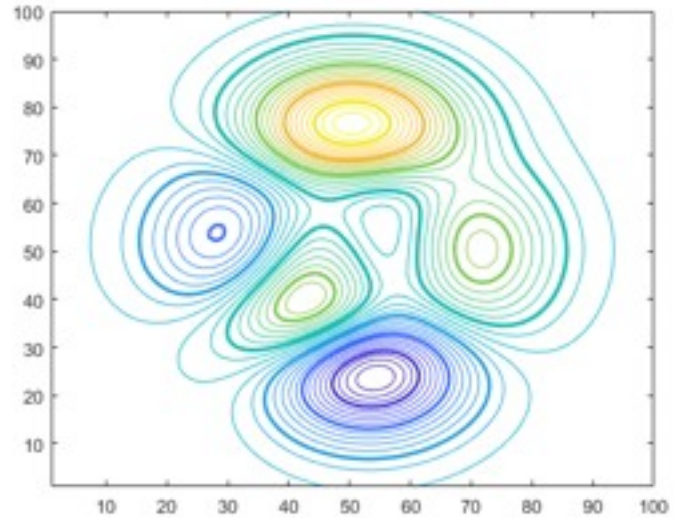


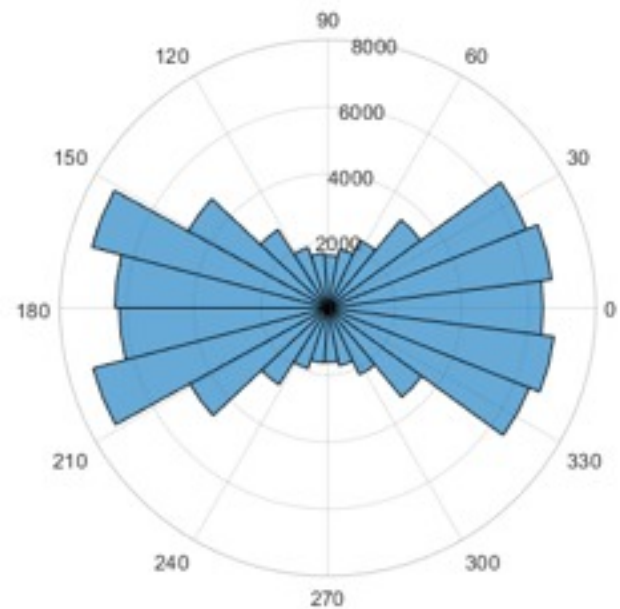
Exkurs: Plots

```
x = linspace(0,10,100)
y = sin(x)
plot(x,y)
```



Exkurs: Plots





- `plot`: 2D plots
- `plot3D`: 3D plots
- `loglog`: Plots mit log. Achsen
- `bar`: Balkendiagramm
- `histogram`: Histogramm
- `pie`: Tortendiagramm
- ...
- https://de.mathworks.com/help/matlab/2-and-3d-plots.html

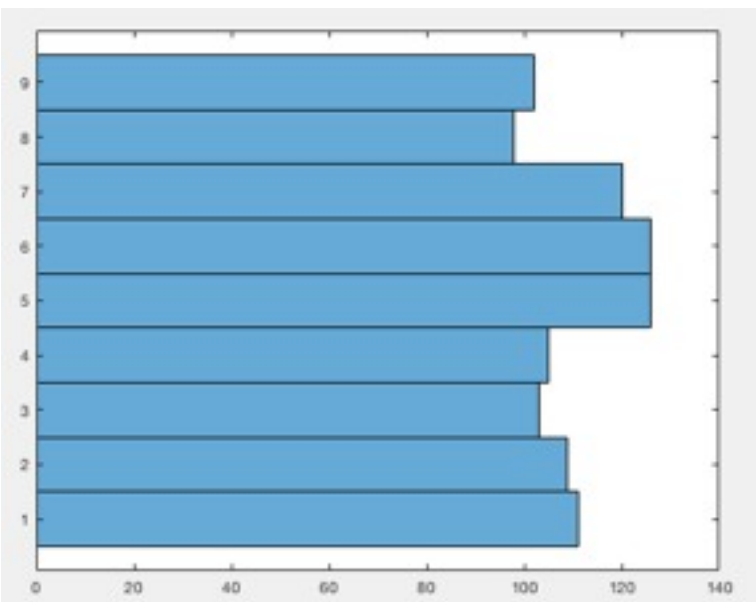
Beispiel: Zufallszahlenverteilung

```
% Vektor mit 1000 Zahlen in ]0,1[ “würfeln”:
zz = rand(1,1000);

zz = zz .* 10;  % -> in ]0,9[
zz = zz .+ 1;   % -> in ]1,11[
zz = floor(zz); % abrunden -> in {1,2,...,10}
```

oder

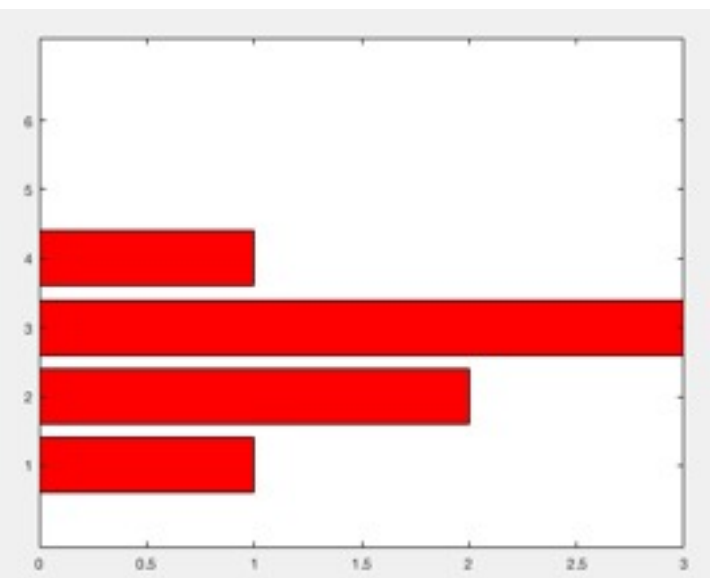
```
randi(10,1,1000)
```

```
% Histogramm generieren (liefert ‘handle’):
h = histogram( zz );

% Eigenschaften ändern:
h.Orientation = 'horizontal';
```



Notenhistogramm (mit Matlab Plot)

```
spektrum = 6;
n = zeros(1,spektrum);
eingabe = 1;

while eingabe ~= 0
    eingabe = input('Note: ');
    if (eingabe > 0) && (eingabe <= spektrum)
       n(eingabe) = n(eingabe) + 1;
    end
end

diagramm = barh( n );
diagramm.FaceColor = [1.0 0 0];
```

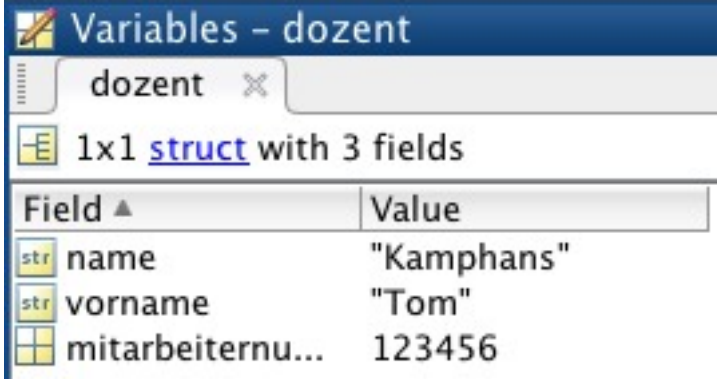


Verbunde/Strukturen

Strukturen (oder Verbunde)

- Container für mehrere Objekte
- Arrays:
  Inhalte werden über num. Indices adressiert
- Strukturen:
  Inhalte (Attribute) werden über Namen adressiert
- Zugriffsoperator . trennt Namen der Strukturvariablen und Namen des Attributs

Strukturen Beispiel




```
dozent.name = "Kamphans";
dozent.vorname = "Tom";
dozent.mitarbeiternummer = 123456;
dozent

  struct with fields:

                  name: "Kamphans"
               vorname: "Tom"
    mitarbeiternummer: 123456

>>fprintf('%s\n', dozent.name);
Kamphans
```

Strukturen Beispiel 2

```
rgb_color.red   = 0;
rgb_color.green = 200;
rgb_color.blue  = 100;

cmyk_color = rgb_to_cmyk( rgb_color );

fprintf('Cyan    Anteil: %5.2f\n', cmyk_color.cyan);
fprintf('Magenta Anteil: %5.2f\n', cmyk_color.magenta);
fprintf('Yellow  Anteil: %5.2f\n', cmyk_color.yellow);
fprintf('Black   Anteil: %5.2f\n', cmyk_color.black);
```

Strukturen Beispiel 2 (Fortsetzung)

```
function cmyk = rgb_to_cmyk( rgb )
  Rtemp = rgb.red   / 255;
  Gtemp = rgb.green / 255;
  Btemp = rgb.blue  / 255;
  K     = 1 - max([Rtemp, Gtemp, Btemp]);
  cmyk.black   = K;
  cmyk.cyan    = (1 - Rtemp-K) / (1 - K);
  cmyk.magenta = (1 - Gtemp-K) / (1 - K);
  cmyk.yellow  = (1 - Btemp-K) / (1 - K);
end
```

```
>> rgb_to_cmyk_script_2
Cyan    Anteil:  1.00
Magenta Anteil:  0.00
Yellow  Anteil:  0.50
Black   Anteil:  0.22
fx >>
```

Kombination mit Arrays

- Erste Indizierung:
  Index des Elementes im Array
- Zweite Indizierung:
  Name des Attributes

```
dozenten(123456).name = 'kamphans';
dozenten(123456).vorname = 'tom';

dozenten(123456)

ans =

  struct with fields:

      name: 'kamphans'
    vorname: 'tom'
```

Abstrakter Datentyp

- Verbund von Daten zusammen mit zulässigen Operationen, die auf sie zugreifen
- Z.B.: Komplexe Zahlen
  - Daten: Realteil, Imaginärteil
  - Operationen: Addition, ..., Betrag, Polardarstellung, ...

Beispiel: Solarmodul

Energieerzeugung
Aus den Werten für Globalstrahlung, Modulneigung, Azimut, Temperaturfaktor und den Eigenschaften des Solarmoduls lässt sich die vom Solarmodul erzeugte Energiemenge E_{ideal} berechnen.

Beispiel:
Solarmodule: 10 x Solarworld SW 250 Mono (Nennleistung STC = 250 W)
Standort: Berlin
Dachneigung: 45°
Azimut: -30° (Süd-Süd-Ost)
Nennleistung PPV: 10 * 250 Watt = 2500 Watt = 2,5 kWp
Globalstrahlung Z_2: 1010 kWh/m² (Mittelwert für Berlin)
Faktor Neigung + Azimut Z_3: 1,1 (aus Diagramm)
Temperaturkorrekturfaktor Z_4: 0,9585

E_{ideal} = PPV * Z_2 * Z_3 * Z_4 * [m² / (kW * Jahr)]
= 2,5 kW * 1010 kWh/m² * 1,1 * 0,9585 = 2662 kWh / Jahr

Beispiel: Solarmodul

```
solarmodul(1).hersteller = 'Solarworld';
solarmodul(1).typ = 'SW 250 Mono';
solarmodul(1).breite = 1675;
solarmodul(1).laenge = 1001;
solarmodul(1).nennleistung = 250;

standorte.berlin.globalstrahlung = 1010;

installation.name = 'HTW Berlin';
installation.anzahl = 10;
installation.dachneigung = 45;
installation.faktor_neigung = 1.1;
installation.temperaturfaktor = 0.9585;

e = Eideal( solarmodul(1), standorte.berlin, installation );
fprintf('Eideal: %5.2f\n', e);

function E = Eideal( modul, standort, installation )
  E = installation.anzahl * ...
      modul.nennleistung / 1000 * ...
      standort.globalstrahlung * ...
      installation.faktor_neigung * ...
      installation.temperaturfaktor;
end
```