Karlsruher Institut für Technologie

MINT-KOLLEG BADEN-WÜRTTEMBERG

Ein Blick hinter die Kulissen
Praxisorientierte Einführung in die Programmierung

Dr. Jürgen Liedtke, 28. Februar 2020

MINT-Kolleg Baden-Württemberg

Begrüßung

Karlsruher Institut für Technologie

MINT-Kolleg Baden-Württemberg

Eine Brücke von der Schule zur Universität

- für ein naturwissenschaftlich-technisches Studium
- mit Angeboten zum Studienbeginn, zur Vorbereitung und
- zur Orientierung in Kooperation mit Schulen (u. a. Mitarbeit in der COSH-Gruppe).

- Ergänzende Online-Angebote in Mathematik und Physik (in Entwicklung),
- verschiedene Workshops für Lehrkräfte sowie Schülerinnen und Schüler (MINT400, Girls' Day, . . .).

Einführung

Programmierkenntnisse erlauben einen eigenen Blick
hinter die Kulissen der digitalen Informationsgesellschaft.

Einführung in die Programmierung

- anhand praktischer Fragestellungen
- in die Formulierung von Anweisungen
- und den Umgang mit Daten: Typen, Ausgabe, Eingabe;
- Vorteile von Feldern
- und Möglichkeiten grafischer Ausgaben und Visualisierungen

selbst erleben.

Zielsetzung

Karlsruher Institut für Technologie

Workshop bietet erste eigene Erfahrungen mit selbst erstellten Programmen für individuelle Anwendungen. Ziele sind:

- Einblick in eine moderne Skriptsprache erhalten
- Grundideen der Programmierung verstehen und anwenden
- moderne Arbeitsumgebung kennenlernen

Übergeordnete Ziele

- Abläufe IT-basierter Arbeitsweisen erfahren
- Bildungspläne: Beispiele für Bezüge zu IMT, NwT, . . . zeigen
- Interdisziplinäres Verständnis fördern, besonders zur Informatik
- Praktische Anforderungen zum Studienbeginn in ing.-wiss. Fächern aufzeigen

Auswahl des Programmierwerkzeugs

Kriterien

- strukturierte Programmierung möglich
- vergleichsweise einfach zu erlernende Interpretersprache
- einfach zu bediendende Arbeitsumgebung
- integrierter Texteditor für Programmcode
- in Technik und Wissenschaft relevante Programmiersprache

Auswahl

- Im Workshop wird mit Octave gearbeitet.
- Industrielle Einsatzmöglichkeiten bietet MATLAB.

Übersicht

GNU Octave

- Gemeinschaftsprojekt, open source-Projekt, siehe `www.octave.org`
- komfortable Arbeitsumgebung
- Interpreter-Sprache (vergleichsweise) einfach zu erlernen
- gute Visualisierungsmöglichkeiten
- kompatible Befehlssyntax zu MATLAB

MATLAB $\hat{=}$ MATrix LABoratory

- entwickelt von Cleve Moler, Universität New Mexico
- professionelles System, siehe `www.mathworks.de`
- viele numerische Berechnungen, speziell für Matrizen
- vielfältige Visualisierungsmöglichkeiten
- umfangreiche Erweiterungen

Themenüberblick

Einführung

Arbeit mit der Programmoberfläche

Eigenschaften und Aufbau eines Programms

Eingabe und Ausgabe von Daten

Programmsteuerung

Ausblick auf andere Skriptsprachen

Leitfaden des Workshops

- Szenario: Beobachtungen eines (begrenzten) Wachstumsverhaltens
- Exemplarisch: Wachstum von Seerosen in einem angelegten Teich oder Bakterien in einer Petrischale
- Beschreibung und Auswertung quantitativer Beobachtungen soll IT-basiert erfolgen
- Fokus des Workshops liegt in der Realisierung der rechnerischen und grafischen Beschreibung

Dazu werden die Konzepte mit alltäglichen Begriffen erläutert (Unterschiede zur Fachterminologie sind zu beachten).

Abschnitt 2

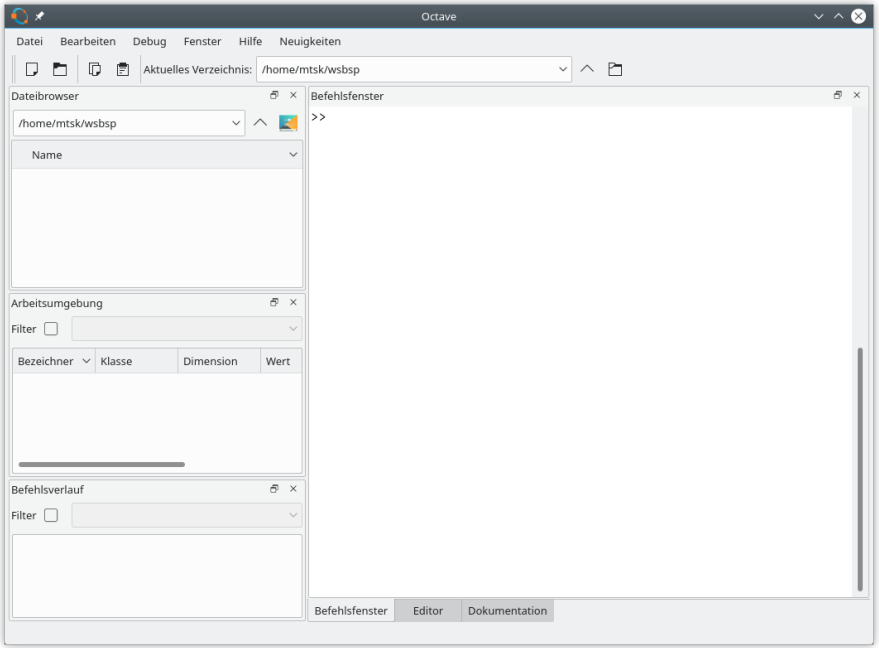
Arbeit mit der Programmoberfläche

- Interaktive Arbeit
- Skripte erstellen

Start der grafischen Benutzeroberfläche

Starten unter

- Windows: Doppelklick auf das Programmicon
- Linux: Programmname auf der Kommandozeile eingeben



Interaktive Arbeit

Einsatz der Konsole

Beispiel: Berechnungen zu einem Bestand (einer Population, . . .):

Elementare Berechnungen

```
» 6 .* 2
» 6 .* 2 .* 2
» A = 6 ./ 24
» 100 .* A
```

Messwerte auswerten

```
» s = 7 + 8 + 4 + 7 + 4
» m = s ./ 5
```

Oder:

```
» x = [7, 8, 4, 7, 4] % Feld von Messwerten
» m = sum(x) ./ length(x) % Einzelwerte summieren und dividieren
```

Interaktive Arbeit II

Grafische Darstellungen

Hinweis: Eingaben können aus der Historie ausgewählt und bearbeitet werden.

Liniengrafik erstellen

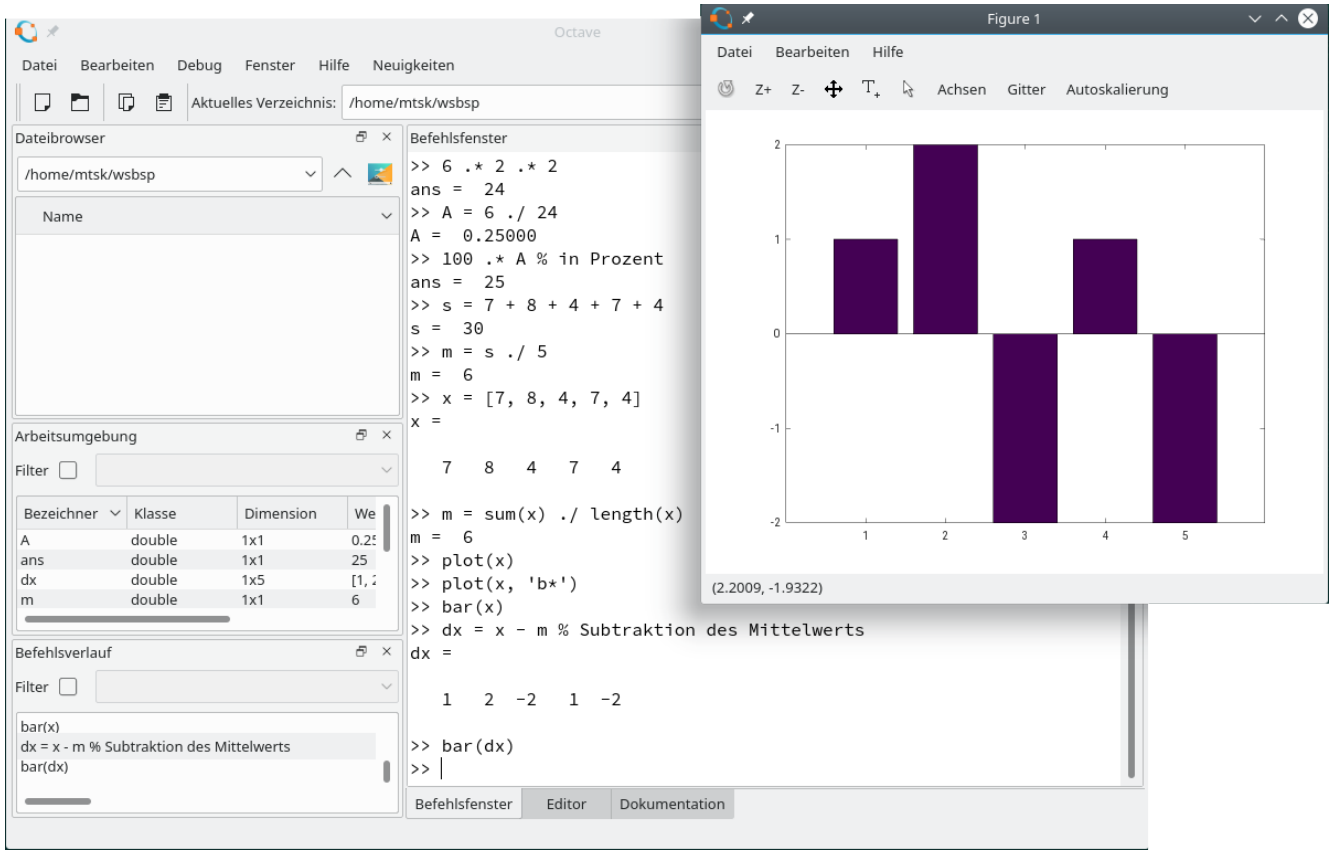
```
» x = [7, 8, 4, 7, 4]
» plot(x)
```

Diagramme erstellen

```
» x = [7, 8, 4, 7, 4]
» m = sum(x) ./ length(x) % Einzelwerte summieren und dividieren
» plot(x, 'b*')
» bar(x)
» dx = x - m % Subtraktion des Mittelwerts
» bar(dx)
```

Interaktive Arbeit II

Arbeitsumgebung und Ergebnisansicht



Abläufe mit Skripten programmieren

Das erste Programm

Skript

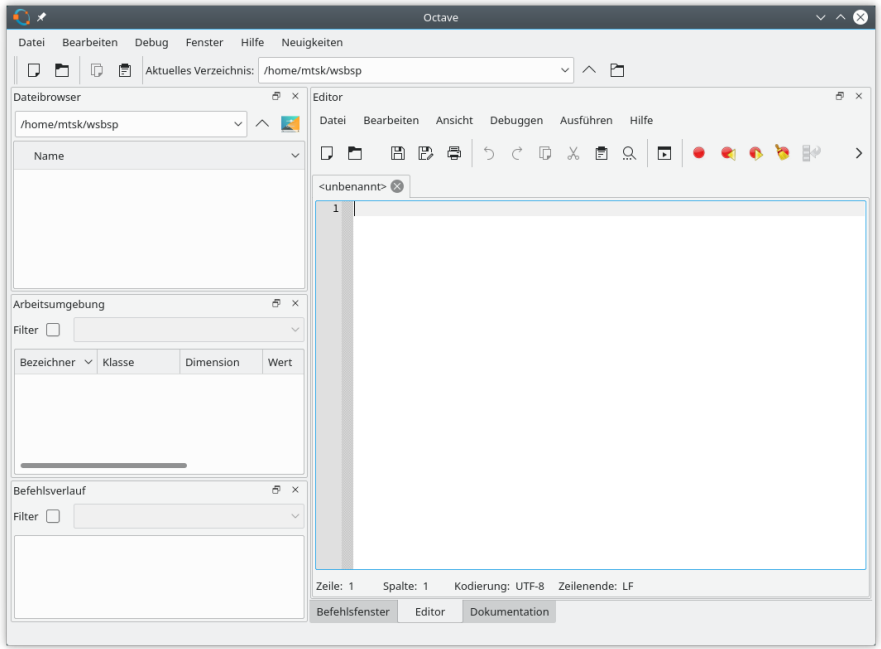
- Abfolge von Kommandos
- gespeichert in einer Textdatei
- mit einem Namen name.m
- vom Interpreter „der Reihe nach“ ausgeführt

Funktion (Prozedur)

- besondere Programmeinheit („Unterprogramm“)
- fasst Befehle zur Bearbeitung einer (Teil-)Aufgabe zusammen

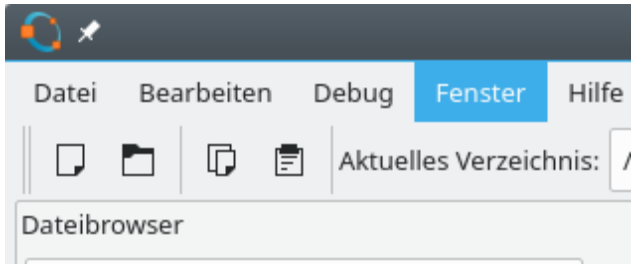
Neues Skript(fenster) öffnen

Editor für das erste Programm aufrufen

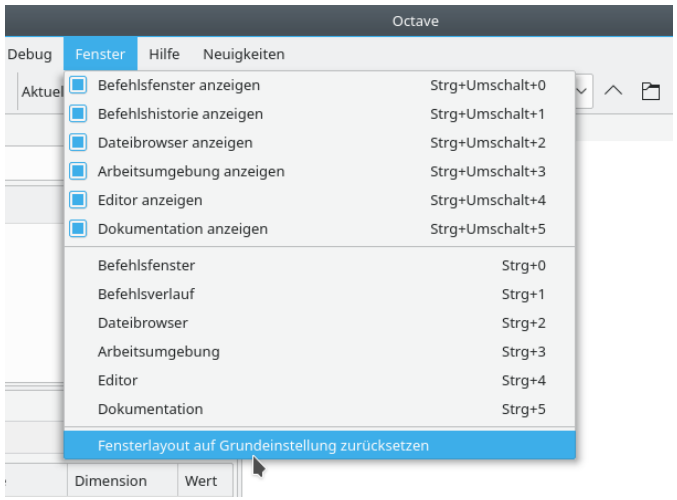


Wo sind die Fenster?

Einstellung der Ansicht der Oberfläche



1. (Haupt-)Fenster ausreichend breit ziehen.
2. Menü `Fenster` öffnen.



3. Funktion zum Fensterlayout ausführen.

Skript schreiben

Das erste Programm

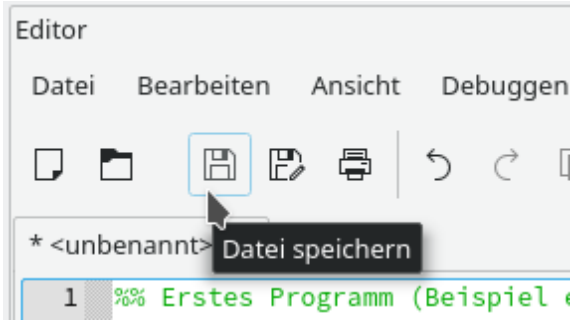
```
%% Erstes Programm (Beispiel eines Skripts)
clc; % Konsole (Befehlsfenster) loeschen.
clear all; % Alle Variablen loeschen.
close all; % Alle Fenster schliessen.

%% Variablen
a = 6; % Variable mit einem Wert belegen.
b = -4; % Variable mit einem Wert belegen.

%% Aktionen
Erg = a + b; % Eine Aktion ausfuehren.
```

Skript speichern

Das erste Programm

Skript speichern

- Register `Editor` wählen
- Button `Save` betätigen
- Verzeichnis festlegen
- Namen vergeben



Dateiname eines Skripts

Ein Name aus

- Buchstaben (beginnend),
- optional Ziffern oder Unterstrich
- mit Suffix „.m“

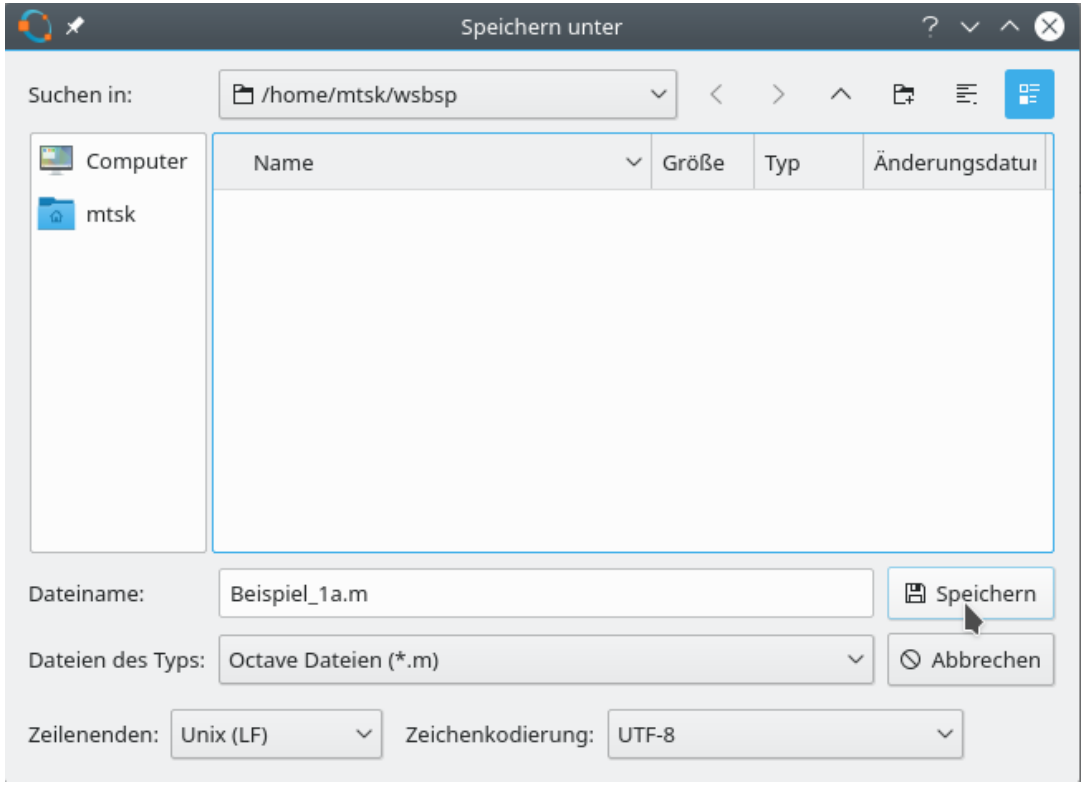
Beispiele

Mögliche Namen sind

- Beispielskript.m
- Bsp1.m
- Beispiel_1a.m

Skript speichern

Das erste Programm



Namen für Skripte

Das erste Programm

Dateinamen vergeben

- mögliche Zeichen: A, . . . , Z, a, . . . z, 0, . . . , 9, _
- Beginn mit einem Buchstaben
- Endung: `.m`
- nicht möglich: Ziffer am Anfang, Leerzeichen, Kommata, Punkte, . . .

Wichtig: Vorhandene Befehle nicht überschreiben!

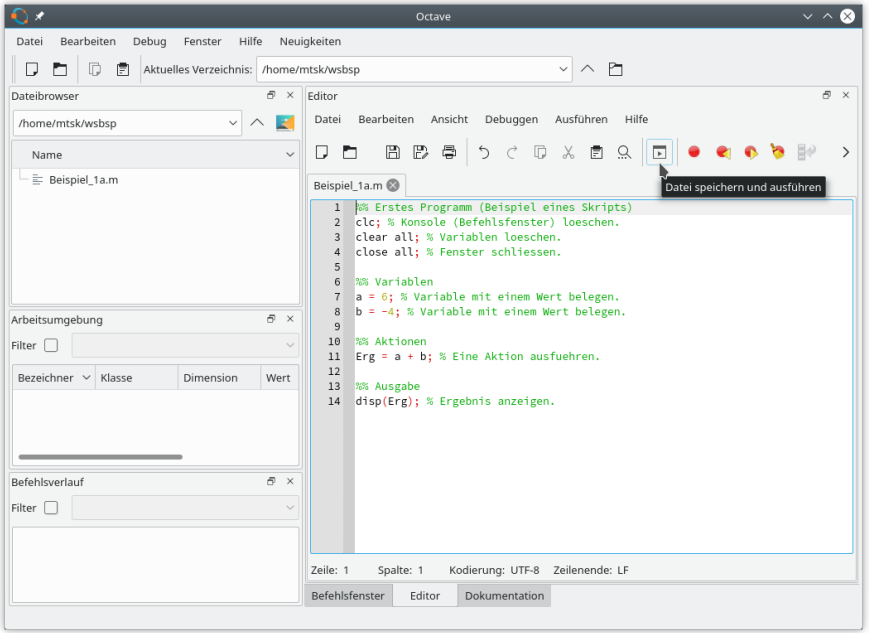
- Beispiel: Eigene Datei sum.m verhindert, dass die Summenfunktion sum verwendet werden kann.
- Lösung: Anderen Namen wählen, zum Beispiel `scSumme.m`.

Skript ausführen

Das erste Programm

Programm ausführen

- Register `Editor` auswählen
- Button `Ausführen` (Run) anklicken

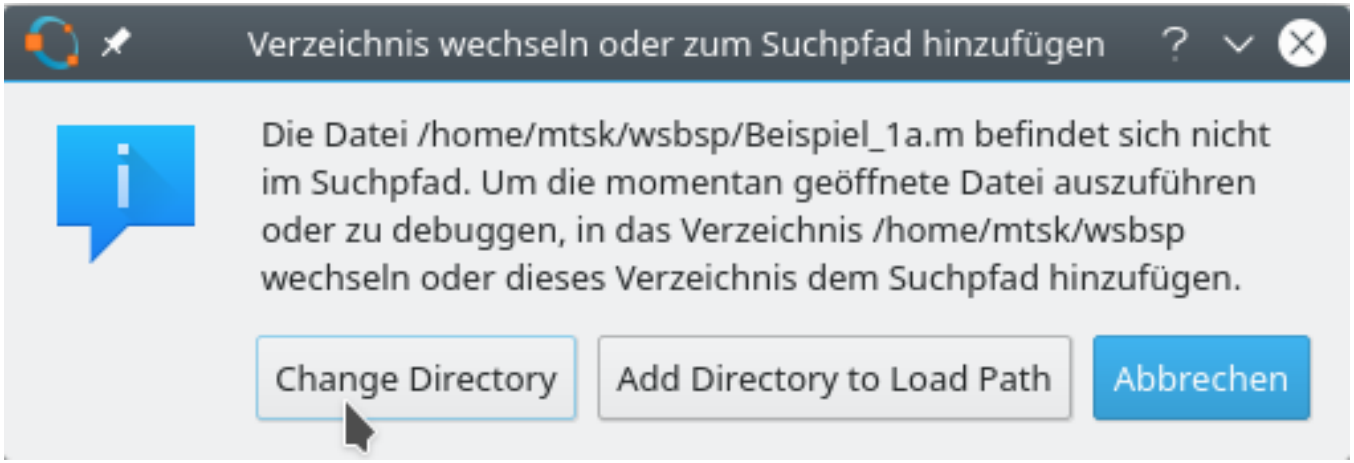


Eventuell ist noch das passende lokale Verzeichnis auszuwählen.

Skript ausführen

Arbeitsverzeichnis festlegen

Beim ersten Start eines Skripts wird dieses oft nicht gefunden:



Betätigen Sie den Button „Change Directory“: Arbeitsverzeichnis wird gewechselt.

Fehlersuche

Beim Programmieren kommt es auch auf die richtige Schreibweise (Syntax) an.

Hinweise zur Fehlersuche

- Suchpfad für Skript aktuell?
- Dateinamen ohne Lehrzeichen geschrieben?
- Namen in Klein- oder Großbuchstaben unterscheiden sich: erg und Erg sind verschiedene Namen.
- Befehl richtig geschrieben?
- Kommentar mit dem Kommentarzeichen % begonnen?
- Dezimalzahl mit Punkt notiert?

Abschnitt 3

Eigenschaften und Aufbau eines Programms

- Programmieren – Was ist das?
- Variablen
- Elementare mathematische Berechnungen und Grafiken

Programmieren – Was ist das?

Einen Ablauf mittels einer (künstlichen) Sprache exakt beschreiben, sodass er automatisch – wie beschrieben – umsetzbar ist.
Ein Ablaufpfad kann grafisch beschrieben werden.

Beispiel

Entwicklung eines Programms

Beispiel: Beschreibung des Wachstumsverhaltens eines Bestandes (einer Population, eines Reaktionsprodukts, einer Teilchensorte, . . .)

Welche Aktionen soll es geben?

Eingabe, Auswertungen (Berechnungen), Ausgabe, Grafik

In welcher Reihenfolge erfolgen die Aktionen?

- Eingabe von Daten
- Berechnungen
- Ausgabe von Ergebnissen
- Grafik zur Visualisierung

Was wird für die Umsetzung des Ablaufs benötigt?

- Objekte
- Aktionen

und eine Ablaufsteuerung mittels

- entscheidungsabhängiger Aktionen (Verzweigungen)
- Wiederholung von Aktionen (Schleifen)

Variablen

Beschreibung von Objekten

Variable: benannter Speicherplatz einer bestimmten Größe, z.B. für

- Zahl 65
- Buchstabe ’A’
- Wort ’Ausgabe’

Variable

- festgelegter Speicherbereich
- mit Name: Wort aus Buchstaben, Ziffernfolge oder Unterstrich
- automatisch angelegt
- Typ vom Interpreter gewählt, wenn nicht explizit angegeben
- Inhalt frei austauschbar

Hinweis: Eine Variable `x` bezeichnet keine gesuchte Größe: Durch die Anweisung `x = x - 6` wird eine Aktion beschrieben.

Datentypen

Beschreibung von Objekten

Zahlen

- Dezimalzahl, z. B. `5`; `-2`; `3.5` (mit Dezimalpunkt notiert)
- komplexe Zahl, z. B. `6 + 4 * i` (mit imaginärer Einheit i)
- Dualzahl (Binärzahl), z. B. `0b10101`, `0B10101`
- Hexadezimalzahl, z. B. 0x4B, 0X4B

Zeichen und Zeichenketten

- Zeichen, z. B. `'A'`, `':'`
- Zeichenkette, z. B. `'Ausgabe'`, `'Zwei Worte'`

Hinweis: Zudem gibt es Konstanten wie `pi` (Kreiszahl).

Datentypen II

Beschreibung von Objekten

Felder von Zahlen

- `s = [7, 8, 4, 7, 4]`
- `s(2) % Zweiter Wert von s`
- `f = [-2, -1, 0, 1, 2; 7, 8, 4, 7, 4]`
- `f(1, 3) % Wert in Zeile 1; Spalte 3 von f`

Felder von Zeichen(ketten)

- `w = ['A', ':']`
- `T = ['Wort', 'schatz'] % moeglich`
- `U = ['Wort', ' und ', 'Satz'] % auch moeglich`

Datentypen III

Beschreibung von Objekten

Automatisch erzeugte Felder von Zahlen

```
F = [Start:Aenderung:Ziel]
```

- `t = [0:1:4], kurz: t = 0:1:4`
- `u = [-3:2:7]`
- `v = [5:-1:2]`

Frage: Was ergibt `f = [3:0.5:7]`, `g = [12:-5:0]`, `h = [-7:0.4:-3]`?

Zusammensetzen von Feldern

- `u = [x, y]`
- `w = 'Wort'; w2 = 'spiele'`
- `Begriff = [w, w2]`

Aktionen mit Zahlen

Berechnungen durchführen

Beispiele

```
a = 6;
b = 2;
p = a .* b
p = p .* b
q1 = 2 ./ 8
q2 = 2 .\ 8
z1 = 3 .* 2 .^ 2
z2 = (3 .* 2) .^ 2
```

Verknüpfungen

- Addition +
- Subtraktion -
- Multiplikation .*
- Division ./ oder .\
- Potenz .^
- Gruppierung ( )

Beispiele zu Verknüpfungen mit Feldern

Berechnungen durchführen

Es wird mit den Feldern `t = [0:1:4]` und `u = [-0.2,0.3,0,-0.4,0.1]` gerechnet:

```
p1 = 3 .* t
p2 = t .* t
p3 = t .^ 2
```

```
q1 = t + 8
q2 = t + 3
```

Varianten mit und ohne Feldverknüpfungen

```
a = 6; b = 2;
y(1) = a
y(2) = y(1) .* b
y(3) = y(2) .* b
y(4) = y(3) .* b
y(5) = y(4) .* b
```

```
t = [0:1:4]
z = a .* b .^ t
```

Modellierung und Funktionen

Berechnungen durchführen II

Frage: Wie kann das Gebiet, eine Kreisfläche, gezeichnet werden?
Funktionale Beschreibung mittels

- x- und y-Koordinaten oder
- Abstand und Winkel

Beispiele zu Funktionen

`sqrt(100)`: Wurzelfunktion, z. B. Abstand von (8, 6) zu (0, 0)
`cos(pi/3)`: Kosinus von $\pi/3$ im Bogenmass
`sind(90)`: Sinus von 90° im Gradmass
`rand(1, 4)`: Feld mit vier Zufallszahlen, jeweils zwischen 0 und 1

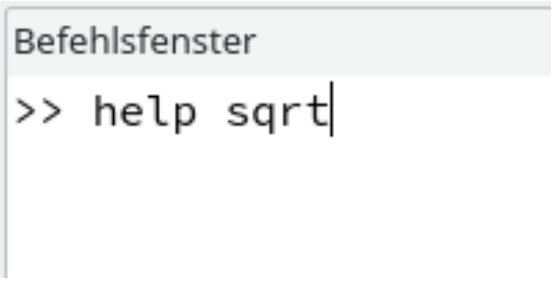
Hinweis zur Dokumentation

- Kurze Information: `help sqrt`
- Dokumentation: `doc sqrt`

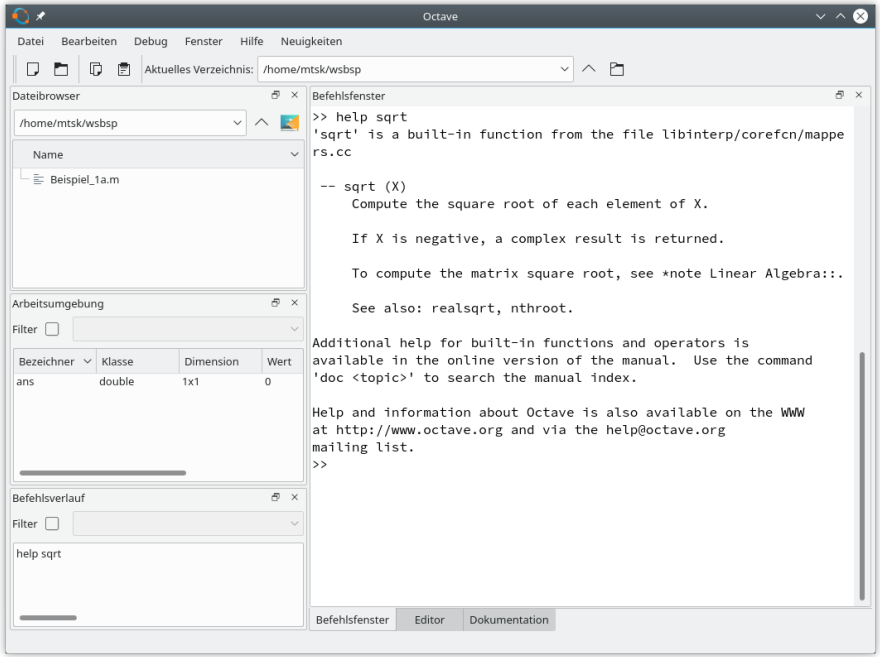
Hilfefunktion

Nutzung der Dokumentation

Beispiel



Hilfetext zur Wurzelfunktion



Dokumentation zu einem Befehl

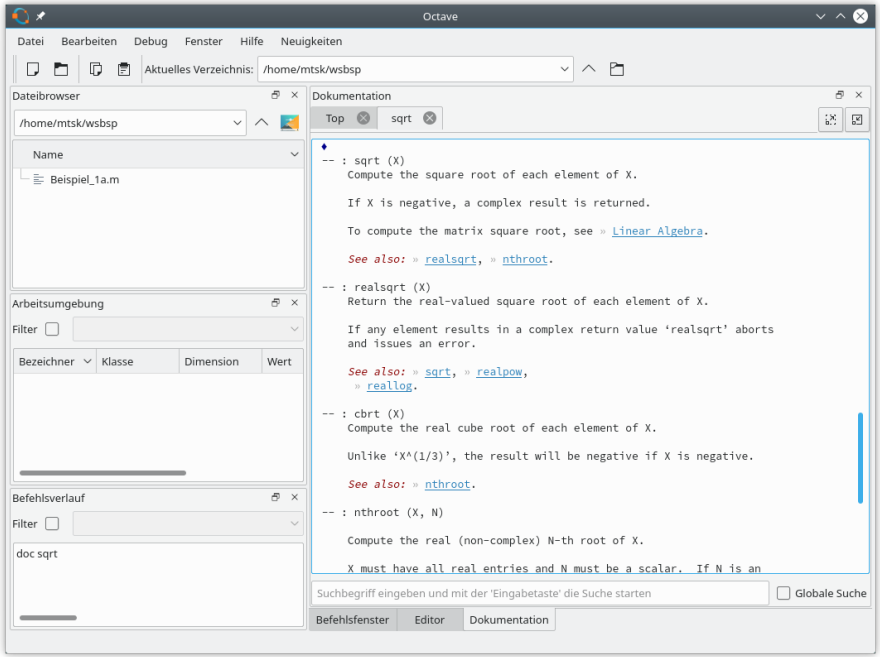
Nutzung der Dokumentation

Beispiel

Befehlsfenster

```
>> doc sqrt
```

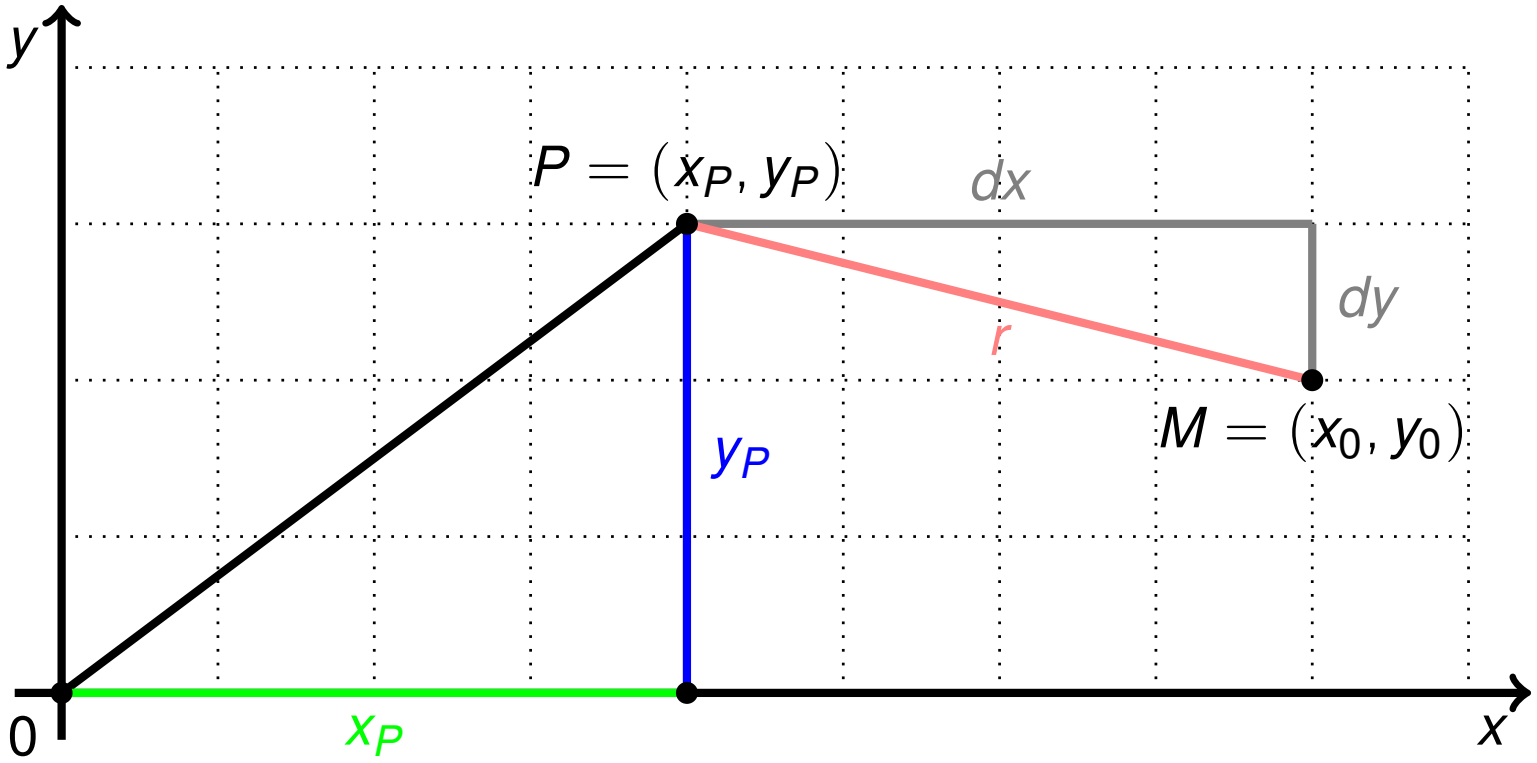
Dokumentation zur Wurzelfunktion



Geländeplan

Koordinaten zur Ortsbeschreibung

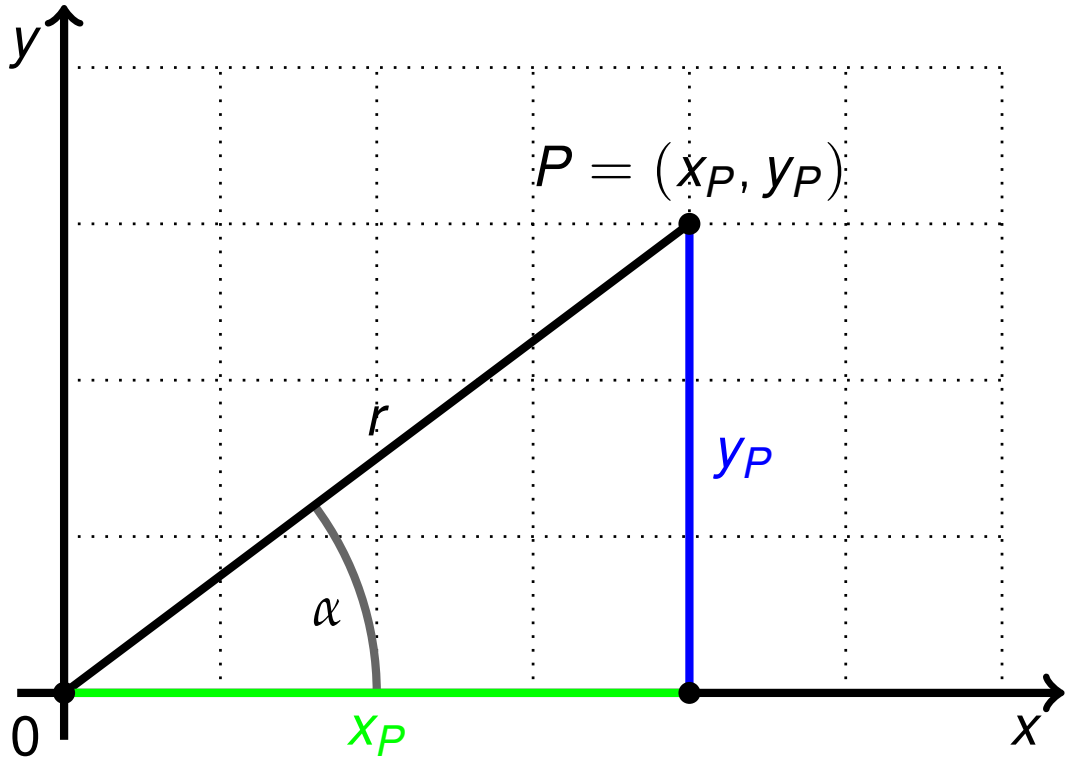
Ideen: Orte wie im Stadtplan beschreiben – und beide Achsen numerisch skalieren



Abstandsquadrat von M zu P ist $r^2 = dx^2 + dy^2$

Neue Koordinaten

Orte mit Abstand und Winkel beschreiben



Umrechnung: $x_P = r \cdot \cos(\alpha)$, $y_P = r \cdot \sin(\alpha)$,

Konstruktion eines Kreises

Beispiel zu Funktionen

Kreis mit Radius *r* als Polygon (näherungsweise) zeichnen: Zu Winkeln, z. B. 0, 6, 12, . . . , 360 Grad, werden Punkte auf dem Kreis mittels Strecken verbunden.

Beispiel eines Kreises um (0,0)

```
alpha = [0:6:360];
r = 5;
x = r .* cosd(alpha);
y = r .* sind(alpha);
plot(x, y);
```

Beobachtungen

Wie sieht die gezeichnete Figur aus?
Was sollte erscheinen?

Grafik

Visualisierung und Hilfsmittel zur Auswertung

Grafikfenster

figure(n)	Grafikfenster öffnen (mit Nummer n)
hold on	im selben Fenster weiterzeichnen
grid on	Koordinatenraster einzeichnen
axis equal	Achsen gleich skalieren
axis([x1,x2,y1,y2])	*x*-Skala: x1 bis x2, *y*-Skala: y1 bis y2

Zeichenbefehle

plot	Liniengrafik zeichnen (oder „Punktgrafik“)
fill	Figur zeichnen (oder: Bereich ausfüllen)
bar	Balkendiagramm

Grafiken erstellen

Grafikfenster einrichten

```
figure(1); % Grafikfenster 1 oeffnen
hold on;
grid on;
title('Entwicklung');
xlabel('Zeit');
ylabel('Anzahl');
```

Beispiel: Funktionsgraph der Bestandsentwicklung

```
t = [0:1:4];
z = 6 .* 2 .^ t;
plot(t);
plot(t, z); axis([0, 5, 0, 100]);
```

Beispiel: Besiedelung zeichnen

Kreis zeichnen

```
% Besiedelungsgebiet (Kreis) zeichnen:
figure(1); axis equal; axis([0, 10, 0, 10]); hold on;
phi=[0:6:360]; x=5 .* cosd(phi) + 5; y=5 .* sind(phi) + 5;
fill(x, y, 'b', 'EdgeColor', 'k');
% Aktuelle Besiedelung zeichnen (ueberall moeglich?):
bx = 10 .* rand(1, 96); by = 10 .* rand(1, 96);
plot(bx, by, 'r*');
```

Beobachtungen:

- Wichtige Informationen zum Programmablauf stehen nur im Quelltext.
- Variationen eines Wertes wie des Radius erfordern hier jeweils eine Änderung des Programms.

Abschnitt 4

Karlsruher Institut für Technologie

Eingabe und Ausgabe von Daten

- Ausgabe von Daten
- Eingabe von Daten

Interaktive Programme

Eingabe und Ausgabe von Daten

Zu der Konstruktion des Siedlungsgebiets im letzten Beispiel ergeben sich Fragen nach Varianten:

- Wie kann über Aktivitäten des Programms informiert werden?
- Auf welche Weise kann die Größe des Gebiets während des Programmablaufs mit eigenen Werten selbst festgelegt werden?

Ziel: Programme sollen allgemeine Lösungsverfahren anbieten, die für verschiedene Daten genutzt werden können.
Dazu gibt es Funktionen zur Interaktion:

- Kommunikation über Bildschirm und Tastatur
- Datenaustausch über Dateien
- Nutzung von Netzwerken

Hier wird die erste Möglichkeit besprochen.

Ausgabe

Ausgabe im Kommandofenster

Informationen zum Programmablauf:
`disp` Ausgabe von Daten (Zahlen, Zeichenketten)

Ausgabe von Daten

- Ausgabe von Ergebnissen
- Informationen zum Zweck des Programms
- Erläuterungen zur Arbeitsweise
- Hinweise zur Bedienung

Beispiel

```
Laenge = 4.16; Breite = 3; A = Laenge .* Breite;
disp('Programm zur Berechnung von Flaechen');
disp('Flaechenformel fuer ein Rechteck: A = Laenge * Breite');
disp(['Der Flaecheninhalt ist ', 'A = ', num2str(A, "%.1f")]);
```

Ausgabe in einem Fenster

Mitteilungsfenster verwenden

`msgbox` Ausgabefenster für Daten (Zeichenketten)

Beispiel

msgbox('Alle Daten zur Berechung sind vorhanden.', 'Information');

Beispiel zur Ausgabe von Zahlen

Laenge = 4; Breite = 3;
mtext = ['Eingabewerte: ', num2str(Laenge), ' und ', num2str(Breite)];
msgbox(mtext, 'Information zu den Eingaben');

Zahlen x werden mit num2str(x) in eine Zeichenkette umgewandelt, damit sie ausgegeben werden können.

Beispiel (mit Formatierung): `num2str(sqrt(2), "%.9f")`

Felder ausgeben

Beispiel zur Datenausgabe

Ausgabe in der Konsole

```
s = [7, 8, 4, 7, 4];
disp('Ausgabe der Messdaten: ');
disp(s);
disp(num2str([5, sqrt(2), 7], "%8.3f") % mit Formatierung
```

Ausgabe in einem Fenster

```
s = [7, 8, 4, 7, 4];
msgbox(['Messdaten: ', num2str(s)]);
```

Eingabe von Daten

Interaktive Programme

Flexible Nutzung des Programms durch individuelle Eingaben ermöglichen

`input`: Eingabe von Werten in der Konsole (mit [Enter] bestätigen)

Anwendungssituationen

- Eingabe von Werten
- Festlegung von Optionen
- Auswahl von Programmfunktionen

Beispiel zur Eingabe von Daten

```
disp('Berechnung der Flaeche eines Kreises');
r = input('Bitte einen Radius eingeben: ');
```

Eingabe von Daten

Interaktive Datenerfassung

input Befehl zur Dateneingabe über die Konsole
inputdlg Dialogfenster zur Dateneingabe

Beispiel zur Eingabe über die Konsole

```
r = input('Bitte einen Radius eingeben: ');
```

Beispiel zur Eingabe mittels Dialogfenster

```
a = inputdlg('Bitte einen Anfangswert eingeben: ');
```

- input auch für Felder geeignet
- inputdlg: oft Umwandlung der Eingabe nötig
- weitere Möglichkeiten zur Datenübergabe: mittels Dateien unterschiedlichster Formate, über ein Netzwerk, über Schnittstellen für Sensoren, . . .

Auswahl von Optionen

Beispiel zur Dateneingabe

listdlg Dialogfenster eines Auswahldialogs

Beispiel

```
% Variable aw: Antwort, stat: Status (Wert 1: Auswahl, 0: Abbruch)
% awlisteca: Liste mit den Auswahlmoeglichkeiten (cell array)
awlisteca = {'Kreis', 'Rechteck', 'Dreieck'};
[aw, stat] = listdlg('ListString', awlisteca, 'SelectionMode', 'single');
```

Weitere Optionen (mit Beispielen)

- Infotext: 'PromptString', 'Figuren:'
- Dialogtitel: 'Name', 'Wahl einer Option'
- Name des Cancel-Buttons: 'CancelString', 'Abbruch'

Beobachtung

Soll ein aktionsabhängiger Programmablauf erreicht werden, sind zudem Kontrollstrukturen erforderlich (siehe Abschnitt 5).

Abschnitt 5

Programmsteuerung

- Verzweigung – Entscheidungsabhängige Wege
- Schleifen – Wiederholung von Aktionen
- Zusammenfassung zu Programmstrukturen

Ablaufsteuerung

Idee: Der Ablauf soll situationsabhängig erfolgen, indem

- verschiedene Anweisungen (Verzweigungen)
- Anweisungen mehrmals (Wiederholungen, Schleifen)

ausgeführt werden.
Dazu werden relevante Eigenschaften abgefragt und bewertet.

Bedingungen

- Bedingung: Abfrage einer Eigenschaft in einer Situation
- Formulierung mittels Vergleichsoperatoren und logischer Verknüpfungen
- Auswertung nach bestimmten Regeln, ob eine Bedingung erfüllt („wahr“) ist, oder nicht („falsch“).

Bedingungen

Beispiele

Beispiele logischer Aussagen zur Formulierung von Bedingungen sind Vergleiche und Verknüpfungen davon.

Einfache Vergleiche

x = 3; y = 7;

größer	x > 5
gleich	x == (y - 4)
kleiner	'A' < 'a'

Verknüpfungen

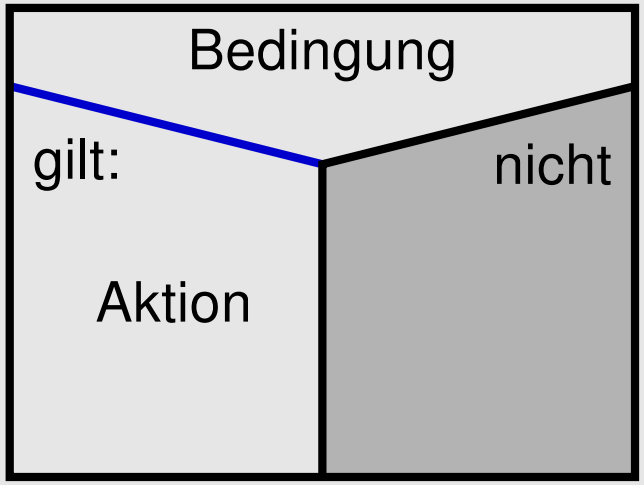
UND	(y > 4) && (x < 1)
ODER	(x == y) || (y < 9)
NICHT	~('Wort' == 'Satz')

Verzweigung

Idee: Je nach Situation sollen Anweisungen ausgeführt werden.

Struktur einer einseitigen Verzweigung

Ein Block von Anweisungen wird nur ausgeführt, wenn eine zuvor geprüfte Bedingung erfüllt wird.



Beispiel

```
if (Winkel > 6)
  disp('Exakte Formel verwendet.');
  y = sind(Winkel);
end
```

Syntax if-Abfrage

```
if logischerAusdruck
  Anweisungen;
end
```

Beispiel

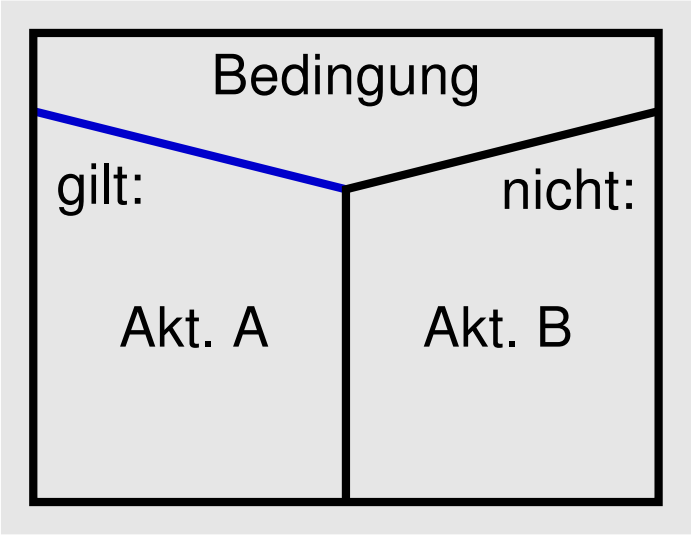
Einseitige if-Verzweigung

```
z = input('Geben Sie eine positive Zahl ein: ');
if (z > 0)
 disp('Eingabe ist positiv.');
end

a = 6;
b = 2;
disp('Eingabe mit [Enter]-Taste bestaetigen: ');
if (input('Kontrolle der Werte [ja/nein]? ') == 'ja')
  disp(['Start: ',num2str(a), ' Basis: ',num2str(b)]);
end
```

Verzweigung II

Struktur einer zweiseitigen Verzweigung

Ein Block von Anweisungen wird ausgeführt, wenn eine zuvor geprüfte Bedingung erfüllt wird, sonst ein anderer Block.



Beispiel

```
if (Winkel > 6)
  y = sind(Winkel);
else
  y = Winkel ./ 60;
end
```

Syntax if-Verzweigung

```
if logischerAusdruck
  Anweisungen;
else
  Anweisungen;
end
```

Beispiel

Zweiseitige if-Verzweigung

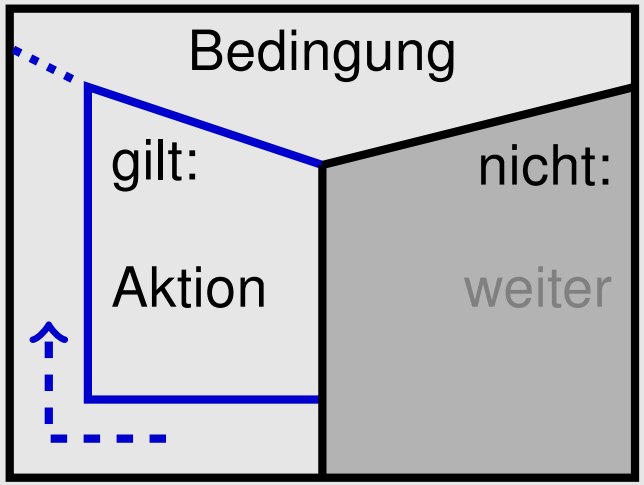
```
a = 6;
a = input('Geben Sie einen Startwert ein: ');
if (a < 0)
  disp('Startwert ist negativ.')
else
  disp(['Berechnung zum Wert ', num2str(a), ':']);
  z = a .* 2 .^ 4;
  disp(['Ergebnis: ' num2str(z)]);
end
```

Wiederholungen: while-Schleifen

Ziel: Ein gewisser Programmabschnitt soll mehrfach ausgeführt werden (solange eine Bedingung erfüllt ist).

Struktur einer Schleife

Ein Block von Anweisungen wird wiederholt ausgeführt, solange eine Bedingung erfüllt ist.



Beispiel (n = 12)

```
while (n > 0)
  n = n - 1; disp(n);
end
```

Syntax while-Schleife

```
while logischerAusdruck
  Anweisungen;
end
```

Beispiele

while-Schleife

```
% Beispiel 1:
n = 1;
n = input('Geben Sie n ein (1 <= n <= 15): ');
while (n < 1) || (n > 15)
  disp('Zahl nicht im richtigen Bereich.');
  n = input('Geben Sie n ein (1 <= n <= 15): ');
end

% Beispiel 2:
figure(1); axis([0, 7, 0, 4]); hold on; k = 1;
while (input('Abbruch mit 6') ~= 6)
  plot([7 .* rand(1,k)], [4 .* rand(1,k)], '*');
  k = k+1;
end
```

Anwendungsbeispiel

Szenische Darstellung mittels while-Schleife

```
clc; clear all; close all;
a = 6; b = 2;
x2 = 4; y2 = 3;
k = 0; kmax = 12;
figure(1); axis([0, x2, 0, y2]); axis equal; hold on;
t = [0:1:kmax];
z = a .* b .^ t;
m = z(1); % Bestand zu t = 0.
while (k <= kmax)
  plot([x2 .* rand(1, m)], [y2 .* rand(1, m)], '*');
  k = k + 1;
  m = z(k + 1) - z(k);
  pause(3);
end
```

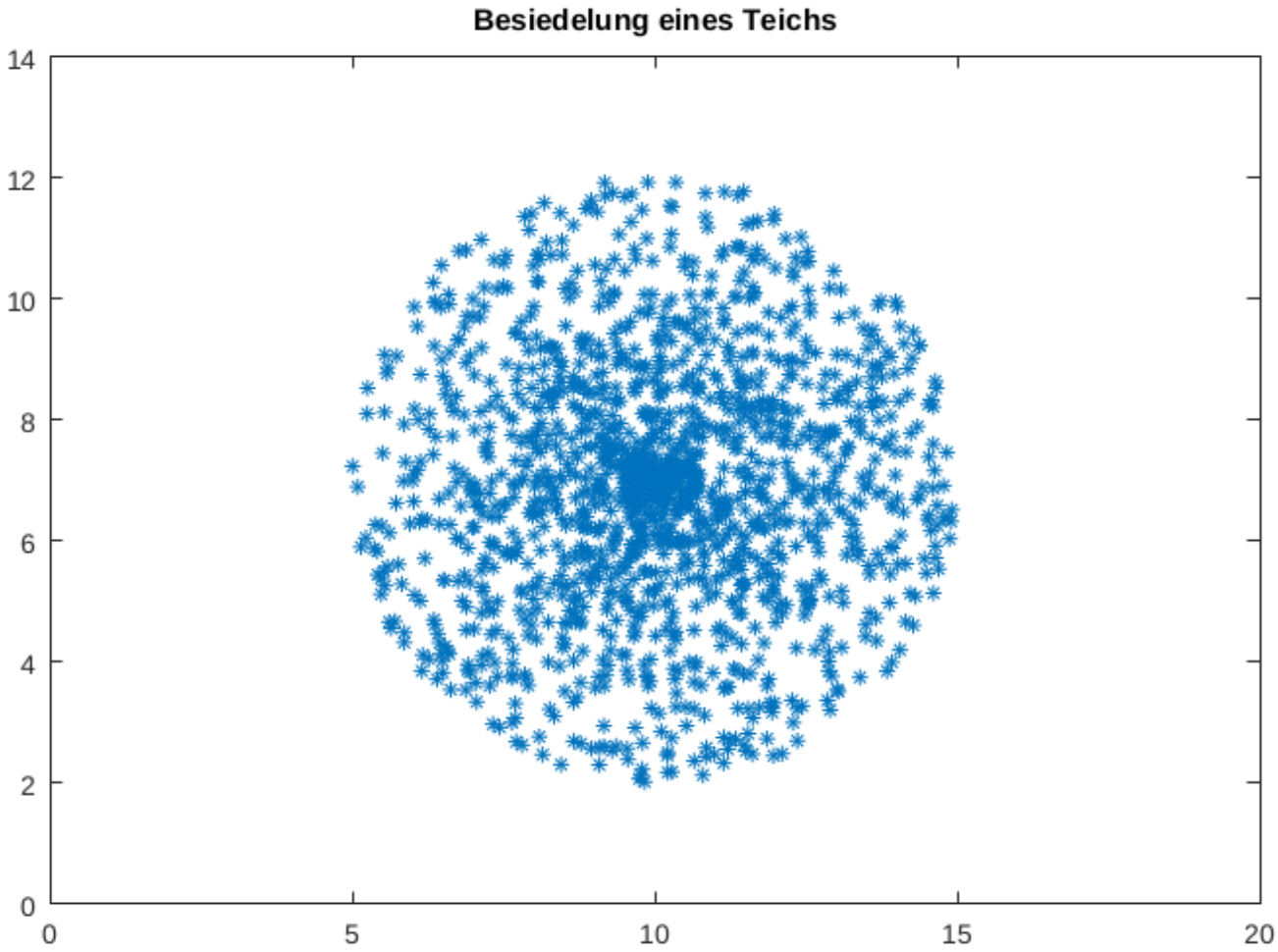
Anwendungsbeispiel

Besiedelung mit rundem Teich

```
clc; clear all; close all;
x2 = 20; y2 = 14;
a = 6; b = 2; kmax = 8;
t = [0:1:kmax];
z = a .* b .^ t;
r = 5 .* rand(1, z(kmax+1));
alpha = 360 .* rand(1, z(kmax+1));
x = r .* cosd(alpha) + 10;
y = r .* sind(alpha) + 7;
figure(1, 'name', 'Bild');
plot(x, y, 'b*'); % blaue Sterne
axis equal; axis([0, x2, 0, y2]);
title('Besiedelung eines Teichs');
```

Anwendungsbeispiel

Bild eines Programmlaufs



Überblick zur Programmstrukturen

Verzweigung mittels if-Anweisung

- Einseitige Verzweigung:
  if [Bedingung] [Anweisungen;] end
- Zweiseitige Verzweigung:
  if [Bedingung] [Anweisungen;] else [Anweisungen;] end

Wiederholung mittels while-Schleife

while [Bedingung] [Anweisungen;] end

Anmerkung

Es gibt weitere Strukturen für Verzweigungen mit mehreren Auswahlen und auch andere Arten von Schleifen.

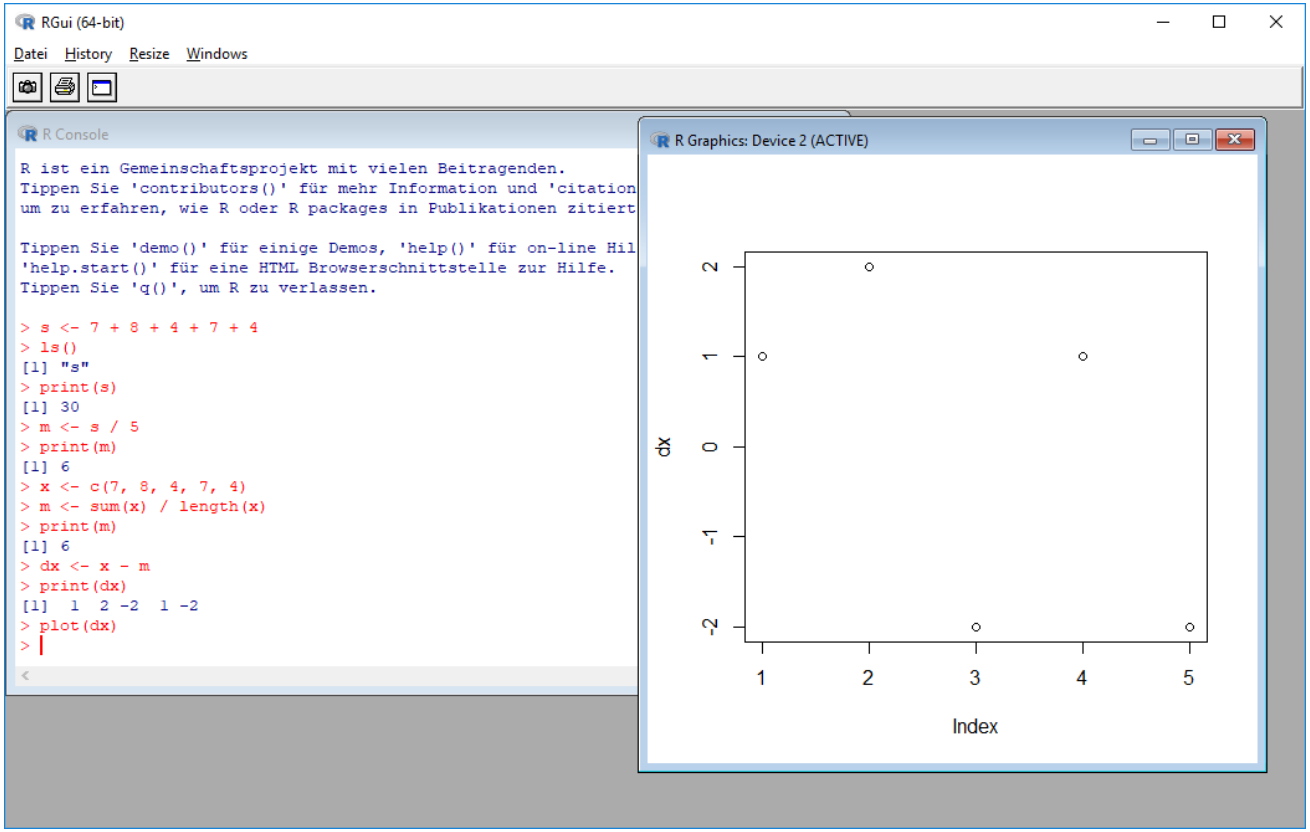
Abschnitt 6

Ausblick auf andere Skriptsprachen

Ausblick auf andere technische Skriptsprachen

Skriptsprache von R

Ansicht von RStudio mit einem Beispiel



Skriptsprache von R

Beispiel

```
x <- c(7, 8, 4, 7, 4)
m <- sum(x) / length(x)
dx <- x - m
plot(dx)
```

Anmerkung

- R bietet besonders Pakete für statistische Fragestellungen,
- Octave und MATLAB für numerische Berechnungen, besonders mit Matrizen.

Literatur und Quellen

- Adlung, Hopp, Köthe, Schnellbächer, Staufer: Tutorium Mathe für Biologen, Springer Verlag, Berlin, 2014.
- Bildungsplan NwT (BW) – Naturwissenschaften und Technik, 2016, allgemeinbildendes Gymnasium, Klassenstufen 7 bis 10.
- Homepage von GNU Octave: `www.octave.org`
- Homepage der Firma MathWorks Inc.: `www.mathworks.com`
- Homepage zu R: `www.r-project.org`
- Wolfgang Schweizer: MATLAB kompakt, 5. Auflage, Oldenbourg Verlag, 2013.

Stand der Online-Referenzen: Februar 2020.

Glossar

Skript	Zusammenfassung von Aktionen
Variable	Speicherbereich mit einem Namen
Zuweisung	Datenübergabe an eine Variable

Syntax und Anweisungen

Allgemeine Befehle

- # Kommentar: Kommentar
- % Kommentar: Kommentar
- Semikolon ; nach einer Anweisung: ohne Ausgabe in der Konsole
- \# Kommentarzeilen \#: Kommentarzeilen
- \% Kommentarzeilen \%: Kommentarzeilen
- clear all: alle Variablen löschen
- close all: alle Grafikfenster schließen
- clc: Inhalt des Ausgabefensters löschen (clear console)
- pause: Unterbrechung, bis eine Taste betätigt wird
- pause(n): Unterbrechung n Sekunden lang
- [Strg] C: Programm abbrechen (beide Tasten gleichzeitig drücken)
- help name: Erklärung zum Begriff Name
- doc name: Dokumentation zum Begriff Name

Syntax und Anweisungen

Objekte und Verknüpfungen

- [a1, a2, . . . , an]: Feld (Zeilenvektor), $1 \times n$-Feld
- [a1; a2; . . . ; an]: Feld (Spaltenvektor), $n \times 1$-Feld
- [a1, a2, a3; b1, b2, b3]: 2×3-Feld (zwei Zeilen mit drei Spalten)
- [p:dn:q]: Feld mit automatisch erstellten Werten von p bis maximal q
- linspace(a,b,n): Feld mit automatisch erstellten Werten von a bis b
- x(k): k-ter Eintrag aus dem Feld x (Zählung beginnt bei 1)
- length(x): Anzahl Einträge von x

Syntax und Anweisungen

Objekte und Verknüpfungen II

- a = c: Zuweisung von c zu a (Variable a erhält Inhalt von c)
- a + b: Addition
- a - b: Subtraktion
- c .* d: elementweise Multiplikation
- c ./ d: elementweise Division (oder d .\ c)
- b .^ t: elementweise Potenz

Funktionen und Konstante

- abs: Absolutwert, Betrag einer Zahl
- sum: Einzelwerte summieren
- rand(1, n): Feld mit n (pseudo-)zufälligen Werten
- sin: Sinus (mit Eingabe im Bogenmaß)
- sind: Sinus (mit Eingabe im Gradmaß)
- cos: Kosinus (mit Eingabe im Bogenmaß)
- cosd: Kosinus (mit Eingabe im Gradmaß)
- pi: Kreiszahl (Verhältnis Umfang zu Durchmesser eines Kreises)

Syntax und Anweisungen

Eingabe und Ausgabe

- disp: Ausgabe im Konsolenfenster (Befehlsfenster)
- msgbox: Informationsfenster
- input: Eingabe im Konsolenfenster (Befehlsfenster) erwartet
- inputdlg: Eingabedialog
- listdlg: Auswahldialog
- questdlg: Fragedialog
- num2str: Zahlenwert als String darstellen
- str2num: String (Zeichenkette) in ein Zahlenformat übersetzen

Syntax und Anweisungen

Grafik

- figure(n): Fenster mit Nummer n für Grafikausgaben
- hold on: Weiterzeichnen (ohne bisherige Zeichnung zu löschen)
- title: Titel einer Grafik
- grid on: Gitterlinien erstellen
- axis equal: Achsen gleich skalieren
- xlabel, ylabel: Beschriften der x- und y-Achse
- legend: Legende erstellen (z. B.: legend('Anzahl', 'Farbwert'))
- subplot: Grafikfenster unterteilen
- plot(y): Liniengrafik erstellen
- plot(x, y): Liniengrafik erstellen
- plot(x, y, 'option'): Liniengrafik mit der Option 'option'
- fill: Figur zeichnen (bzw. Fläche füllen)
- bar: Balkendiagramm, Histogramm

Syntax und Anweisungen

Vergleiche und Programmsteuerung

- a < b, a == b, a > b, a ~= b: Vergleiche (kleiner, gleich, größer, ungleich)
- A && B, A || B, ~A: logische Verküpfungen (UND, ODER, NICHT)
- if [Bedingung] [Anweisungen;] end
- if [Bedingung] [Anweisungen;] else [Anweisungen;] end
- while [Bedingung] [Anweisungen;] end

Laufendes Skript abbrechen

[Strg] C: Abbruch eines Skripts (beide Tasten gleichzeitig drücken)

MINT-Kolleg online unter

www.mint-kolleg.de

Baden-Württemberg

MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Dem Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg (Strukturmodelle individueller Geschwindigkeit) und dem Bundesministerium für Bildung und Forschung (Qualitätspakt Lehre, FKZ 01PL16018A, 01PL16018B) danken wir herzlich für die finanzielle Unterstützung des MINT-Kollegs, eine Verbundeinrichtung des KIT und der Universität Stuttgart.