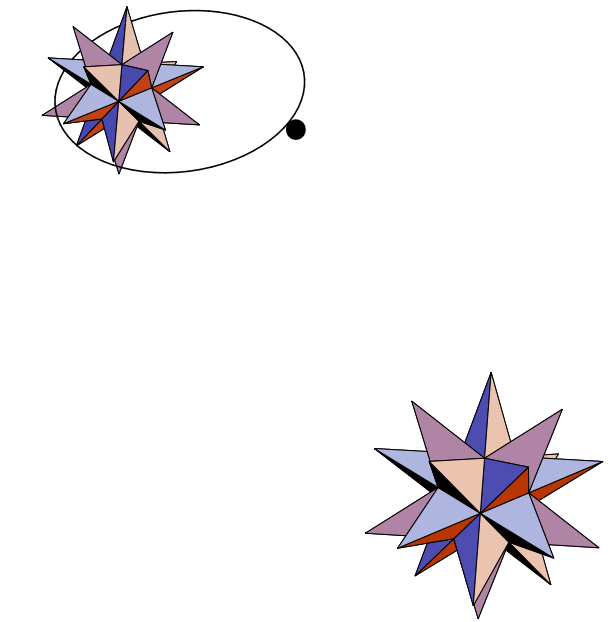
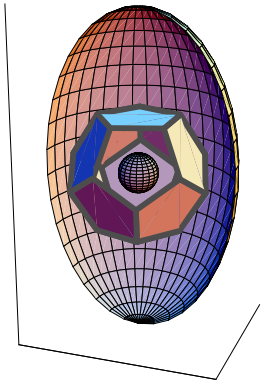


"Kleine Einführung" in *Mathematica*
"Petite Introduction" en *Mathematica*



Ein Kurs zum Selbststudium mit Beispielen und Konzepten nach den Vorbildern "Handbuch Mathematica" von S. Wolfram, "Mathematica: A Practical Approach" von Nancy Blachman und andern Werken.
€ Un cours pour études individuelles avec des exemples et concepts d'après les modèles "Livre de Mathematica" de S. Wolfram, "Mathematica: A Practical Approach" de Nancy Blachman et d'autres oeuvres.

Zusammengestellt von € composé de
Rolf Wirz
Ingenieurschule Biel // école d'ingénieurs Bienne // HTA Biel-Bienne
1993/94/95/98/99

Lauffähige Files für Windows
• Des dossiers - (Files) - exécutables pour Windows

Copyright Rolf Wirz

Inhalt € Contenu

■ Read_me_Kurs

■ I. Rundgang in *Mathematica*: Einstieg und Uebersicht
€ Tour en *Mathematica*: Introduction et vue d'ensemble

(Beispiele nach Wolfram u.a.
• Exemples d'après Wolfram et aautres)

■ II. Kurs: Systematische Einführung in die Sprache
€ Cours: Introduction systématique dans la langue de programmation

(Beispiele nach Blachman u.a., Print gemischt mit III
• Exemples d'après Blachman et aautres, print mêlé à III)

■ III. Uebungen zur systematischen Einführung
€ Exercices pour l'introduction systématique

(Beispiele nach Blachman u.a., Print gemischt mit II
• Exemples d'après Blachman et autres, print mêlé à II)

■ IV. Uebungen zu speziellen Unterrichtsthemen
€ Exercices pour des sujets spéciaux

(Diverse Beispiele • Exemples divers)

■ V. Zur Filestruktur
€ Quant à la structure des files

■ I. Rundgang in *Mathematica*: Einstieg und Uebersicht
€ Tour en *Mathematica*: Introduction et vue d'ensemble

- Uebersicht
  € Vue d'ensemble
- 1. Numerische Rechnungen
  € Calculs numériques
- 2. Graphiken
  € Graphiques
- 3 . Algebra und Analysis
  € Algèbre et analyse
- 4. Gleichungen lösen
  € Résoudre des équations
- 5. Listen
  € Listes
- 6. Matrizen
  € Matrices
- 7. Transformationsregeln und Definitionen
  € Règles de transformation et définitions
- 8. Symbolisches Rechnen
  € Calculs symboliques
- 9. Programmierung
  € Programmation
- 10. Mathematica Packages
  € Packages en *Mathematica*

- **11. Datenaustausch mit Mathematica, andere Ausgabeformen**
  € Echanges de données avec *Mathematica*,

- **Files:**

Einst_00.ma: (Uebersicht • Vue d'ensemble)
Einst_01.ma: (Numerische Rechnungen • Calculs numériques)
etc.
.....
.....
.....
Einst_11.ma: (Datenaustausch mit Mathematica, andere Ausgabeformen
• Echanges de donnés avec *Mathematica*, d'autres possibilités d'output)

■ II. Kurs: Systematische Einführung in die Sprache
€ Cours: Introduction systématique dans la langue de programmation

In diesem Inhaltsverzeichnis ist aus Platzgründen pro File nur die oberste Gliederungsebene wiedergegeben.
• Dans cet index nous n'avons reproduit pour des raisons de place que le niveau supérieur de répaartition par dossier.

- **1. Einstieg in Mathematica**
  € Introduction en *Mathematica*

- **1.1. Maschinenspezifisches**
  € Problèmes spécifiques selon la machine

- **1.2. Versionspezifisches**
  € Problèmes spécifiques selon la version

- **1.3. Front-End versus Kernel**
  € Front-End versus Kernel

- **1.4. Notation**
  € Notation

- **1.5. Mathematica interaktiv benutzen**
  € Utiliser *Mathematica* interactivement

- **1.6. Reserviere Namen**
  € Des noms réservés

- **1.7. Help-Funktion und Kommando-Vervollständigung**
  € Fonction Help et achèvement des commandes

- **1.8. Help-Funktion und Kommando-Vervollständigung bei Zeichen**
  € Fonction Help et achèvement des commandes pour des symboles

- **1.9. Die verschiedenen Klammertypen**
  € Differents tyypes de parenthèses

- **1.10. Packages**
  € Packages

- **File € Dossier:**

Kurs_01.ma

- **2. Numerische Probleme**
  € Problèmes numériques

- **2.1. Arithmetische Operationen**
  € Opérations arithmétiques

- **2.2. Rationale Zahlen**
  € Nombres rationaux

- **2.3. Irrationale Zahlen**
  € Nombres irrationaux

- **2.4. Annäherung durch Dezimalbrüche**
  € Approximation par fractions décimales

- **2.5. Komplexe Zahlen**
  € Nombres complexes

- **2.6. Symbole und Zahlen, Listen etc.**
  € Symboles et nombres, listes etc.

- **2.7. Approximationen**
  € Approximations

- **2.8. Formatierte Zahlenausgabe**
  € Représentations de nombres formatés

- **2.9. Wichtige gespeicherte Konstanten**
  € Constantes mises en mémoire importantes

- **2.10. Zufallszahlen**
  € Nombre probables

- **2.11. Iteratoren**
  € Itérateurs

- **2.12. Matrizenrechnung**
  € Calcul des matrices

- **2.13. Lösen von Gleichungen**
  € Résoudre des équations

- **2.14. Numerische Integration etc.**
  € Intégration numérique

- **2.15. Numerische Lösung von Differentialgleichungen**
  € Solution numérique d'équations différentielles

- **File € Dossier:**

Kurs_02.ma

- **3. Algebraische und symbolische Stärken**
  € Des aventages algébriques et symboliques

- **3.1. Algebra**
  € Algèbre

- **3.2. Gleichungen lösen, Lösung weiterverwenden**
  € Résoudre des équations différentielles, réutiliser les solutions

- **3.3. Vereinfachungen**
  € Simplifications

- **3.4. Summation**
  € Sommation

- **3.5. Calculus (Differential- und Integralrechnung)**
  € Calcul (différentiel et intégral)

- **3.6. Grenzwerte**
  € Valeurs limites

- **3.7. Potenzreihen**
  € Séries de puissances

- **3.8. Lösen von Differentialgleichungen**
  € Résoudre des équations différentiels

- **File € Dossier:**

Kurs_03.ma

- **4. Graphiken**
  € Graphiques

- **4.1. Zweidimensionale Plots**
  € Plots à deux dimensions

- **4.2. Options**
  € Options

- **4.3. Mehrere Graphen in einem Bild**
  € Plusieurs graphiques en un diagramme resp. image

- **4.4. Parametrisierte Kurven**
  € Courbes paramétrisées

- **4.5. Weitere Options**
  € D'autres options

- **4.6. Plots von Flächen im Raum**
  € Plots de surfaces dans l'espace

- **4.7. Options für Farben, Beleuchtung, ...**
  € Options pour couleurs, éclairage, ...

- **4.8. Arbeiten mit Daten**
  € Travail avec des données

- **4.9. Eingebaute Graphikelemente oder Blöcke**
  € Eléments graphiques ou blocs incorporés

- **4.10. Beschriftung von Graphiken (Labels)**
  € Inscriptions sur graphiques (Labels)

- **4.11. Graphik-Pakete**
  € Paquets de graphiques

- **4.12. Animationen**
  € Animations

- **4. 13. PostScript**
  € PostScript

- **4. 14. Experimentieren mit Animationen**
  € Expérimenter avec animations

- **4.15. Probiere eigene Beispiele aus!**
  € Essaie des exemples propres

- **File € Dossier:**

Kurs_04.ma

5. Etwas Umgang mit *Mathematica*

- **5.1. Seiten-Breite setzen**
  € Mettre la largeur d'une page
- **5.2. Listings von Input und Output**
  € Listings d'input et output
- **5.3. Zum Editor**
  € Quant à l'éditeur
- **5.4. Zur On-Line Hilfe**
  € Quant à l'aide on-line
- **5.5. Ein Blank kann "oder" bedeuten...**
  € Un vide peut signifier "ou" ....
- **5.6. Symbollisten erzeugen**
  € Créer des listes symboliques
- **5.7. Verschiedene Möglichkeiten, eine Funktion zu codieren**
  € Différents possibilités de créer des fonctions
- **5.8. Output unterdrücken**
  € Empêcher l'output
- **5.9. Timing - Rechenzeit messen**
  € Mesurer le temps de calculation
- **5.10 Abbruch und Unterbruch**
  € Rupture et interruption
- **5.11 Globale Variablen**
  € Variables globales
- **5.12 Spezielle Formen**
  € Formes spéciales
- **File € Dossier:**

Kurs_05.ma

6. Manipulation von Listen
€ Manipulation de listes

- **6.0. Einleitung**
  € Introduction
- **6.1. Erzeugung von Listen**
  € Création de listes
- **6.2. Umordnen von Listen**
  € Rearranger des listes
- **6.3. Erweiterung und Verkürzung von Listen**
  € Elargir et abréger des listes
- **6.4. Zählung der Elemente einer Liste**
  € Compter les éléments d'une liste
- **6.5. Zusammenfügen von Listen, Beziehungen zwischen Listen**
  € Réunir des listes, relations entre listes
- **6.6. Veränderung der Form einer Liste**
  € Changer la forme d'une liste
- **6.7. Elemente herauspicken**
  € Choisir des éléments
- **6.8. Auswahl von Daten**
  € Choix de données
- **6.9. Rechnen mit Listen**
  € Calculer avec des listes
- **6.10. Auf Listen anwendbare arithmetische Funktionen**
  € Fonctions arithmétiques applicables ä des listes
- **6.11. Listable und Map**
  € Listable et map
- **6.12. Formatierung von Listen**
  € Formater des listes

- **File € Dossier:**

Kurs_06.ma

- **7. Probleme zum Thema "Zuordnungen und Regeln"**
  € Problèmes au sujet de "coordinations et règles"

- **7.1. Zuordnungen**
  € Coordinations

- **7.2. Löschen der Wertzuweisung**
  € Effacer et assigner des valeurs

- **7.3. Regeln**
  € Règles

- **7.4. Gleichheit**
  € Egalité

- **File € Dossier:**

Kurs_07.ma

- **8. Datentypen**
  € Types de données

- **8.1. Atomare Typen**
  € Types atomiques

- **8.2. Andere Typen und Prädikatenfunktionen**
  € Autres types et fonctions de prédicates

- **8.3. Interne Darstellung in *Mathematica***
  € Représentation interne en *Mathematica*

- **8.4. Herauslesen von gewissen Teilen von Ausdrücken**
  € Lirecertains parties d'expressions

- **8.5. Symbole wie "unendlich" etc.**
  € Symboles tels que "infinie" etc.

- **8.6. Memory: Wie Mathematica speichert**
  € Memory: Mise en mémoire d'après *Mathematica*

- **File € Dossier:**

Kurs_08.ma

- **9. Funktionen definieren**
  € Définir des fonctions

- **9.1. Einfache Funktionen**
  € Fonctions simples

- **9.2. Typenprüfung**
  € Examens de types

- **9.3. Bedingte Ausführung**
  € Exécution conditionnée

- **9.4. Vorgegebene Werte**
  € Valeurs données

- **9.5. Die Abarbeitungshierarchie bei Regeln**
  € Hierarchie d'élaboration pour règles

- **9.6. Zusammenfügen von Regeln mit "f/: "**
  € Réunir des règles par "f/: "

- **9.7. Dokumentieren der eigenen Funktionen**
  € Documenter les propres fonctions

- **9.8. Attribute**
  € Attributs

- **9.9. Die Art der Abarbeitung eines Ausdrucks**
  € La façon

- **9.10. Diskrete Funktionen**
  € Fonctions discrètes

- **File € Dossier:**

Kurs_09.ma

- **10. Lokale Variablen und prozedurales Programmieren**
  € Variables locales et programmation "procédurales"

- **10.1. Globale Variablen**
  € Variables globales

- **10.2. Lokale Variablen im Unterschied zu globalen Variablen**
  € Variables locales à la différence des variables globales

- **10.3. Prozedurale Programmierung**
  € Programmation procédurale

- **File € Dossier :**

Kurs_10.ma

- **11. Problemsammlung zur Musterentsprechung**
  (Mustererkennen, musterkonformes Abarbeiten)
  € Collection de problèmes pour "la corréspondence de patrons"
  (Reconnaître des patrons, élaborer selon les patrons

- **11.1. Mustererkennung bei einer Sequenz**
  € Reconnaître les patrons d'une séquence

- **11.2. Nachbau von Funktionen**
  € Reconstruire des fonctions

- **11.3. "Polymorphe" Definitionen**
  € Définitions "polymorphes"

- **11.4. Benennung von Ausdrücken**
  € Nommer des expressions

- **11.5. Ausdrücke auffinden, die mit Mustern übereinstimmen**
  € Trouver des expressions qui s'accordent aux patrons

- **11.6. Das Attribut "Orderless"**
  € Lattribut "Orderless"

- **11.7. Beispiele mit Mustererkennung**
  € Exemples avec reconnaissance de patrons

- **File € Dossier:**

Kurs_11.ma

- **12. Anonyme Funktionen**
  € Fonctions anonymes

- **12.1. "Function"**
  € "Function"

- **12.2. Anwendung auf Datenselektion**
  € Utiliser pour la sélection des données

- **12.3. Neuauflage einer früher definierten Funktion**
  € Nouvelle édition d'une fonction définie auparavant

- **12.4. Transformation von Wertepaaren in Regeln**
  € Transformation de paires de valeurs en règles

- **12.5. Mehrere Argumente**
  € Plusieurs arguments

- **12.6. Daten filtern**
  € Filtrer des données

- **File € Dossier:**

Kurs_12.ma

- **13. Fallstricke und "Debugging" (Fehler eliminieren)**
  € Pièges et "Debugging" (éliminer les fautes)

- **13.1. Fehlermeldungen**
  € Rapport d'erreurs

- **13.2. Fehler durch vordefinierte Variablen**
  € Erreurs par des variables prédéfinies

- **13.3. Unvollständige Befehle**
  € Ordres incomplets

- **13.4. Falsche Klammerung**
  € Parenthèses mal utilisées

- **13.5. Verwechslung von Zeilen- und Spaltenvektor**
  € Confusion entre vecteurs de lignes et de colonnes

- **13.6. Weiter mit "Return" statt "Enter"**
  € Continuer par "Return" au lieu de "Enter"

- **13.7. Probleme mit Wurzeln (Mehrdeutigkeiten)**
  € Problèmes de racines (interprétations différentes, équivoques)

- **13.8. Abarbeitungsreihenfolge**
  € Suites d'élaborations

- **13.9. Ordnung von Regeln**
  € Ordre de règles

- **13.10. "Protect" und Funktionen**
  € "Protect" et fonctions

- **13.11. Debugging**
  € Debugging

- **File € Dossier:**

Kurs_13.ma

- **14. Input und Output**
  € Input et output

- **14.1. Input**
  € Input

- **14.2. Datenexport**
  € Exportation de données

- **14.3. Manipulation von Strings**
  € Amnipulation de strings

- **14.4. Eingabe- und Ausgabeform**
  € Formes d'entrées et de sorties

- **14.5. Uebersetzen von Ausdrücken in andere Programmiersprachen und Manipulation von Sourcecode**
  € Traduire des expressions en d'autres langues de programmation et manipulation du sourcecode

- **14.6. Formate**
  € Formats

- **File € Dossier:**

Kurs_14.ma

- **15. Packages**
  € Packages

- **15.1. Wieso Packages? Wie zugreifen?**
  € Courquoi des packages? Comment les saisir

- **15.2. Contexts**
  € Contexts

- **15.3. Context-Wechsel**
  € Changement de contexte

- **15.4. Zum Laden von Packages**
  € Pour charger des packages

- **15.5. Der Package-Stil**
  € Le style-package

- **15.6. Namenskonflikte**
  € Conflicts de noms

- **File € Dossiers:**

Kurs_15.ma

■ III.Uebungen zur systematischen Einführung
€ Exercices pour l'introduction systématique

- **Files € Dossiers:**

Uebung01.ma
Uebung02.ma
Uebung03.ma
Uebung04.ma
Uebung05.ma
Uebung06.ma
Uebung07.ma
Uebung08.ma
Uebung09.ma
Uebung10.ma
Uebung11.ma
Uebung12.ma
Uebung13.ma
Uebung14.ma
Uebung15.ma
Dazu existieren diverse Daten- und Programmfiles, die der Ordnung im Directory
und der Lesbarkeit wegen alle mit "AAA" oder mit "aaa" beginnen. Man findet sie daher zuoberst.
• Il existe aussi différents dossiers de données et de programmes qui commencent tous par "AAA"
ou par "aaa" pour des raisons d'ordre dans le directoir et pour une meilleure lisibilité. On les trouve
tout en haut.

■ IV. Uebungen zu speziellen Unterrichtsthemen
€ Exercices sur des thèmes spéciaux

- **Es existieren Uebungen zu diversen Themen. Der Themenkreis wird ständig erweitert. Eine lückenlose Auflistung der Files ist hier nicht möglich. Die Filenamen beginnen jeweils mit "Ueb_"**
  € Il existe des exercices sur différentes thèmes.La nombre de thèmes s'élargit constamment. Une liste complète des dossiers n'est pas possible. Les noms des dossiers commencent par "Ueb_"

- **Files € Dossiers:**

Ueb_1_Thema.ma
etc.
.....
.....
.....
Ueb_xx_Thema.ma

■ V. Zur Filestruktur
€ Auant à la structure des dossiers

▪ **Die Folder sind wie folgt gegliedert:**
€ Les directoires sont réparties de la façon suivante:

Eigenes "Root" -> "MatheFiles" -> "Daten"
-> "Kurs"
-> "Uebungen"
-> "KopieDerDaten"
(Sicherung gegen Datenverlust) • Propre "Root" -> "MatheFiles" -> "Daten"
-> "Kurs"
-> "Uebungen"
-> "KopieDerDaten"
(Mesure de sécurité contre la perte des données)

Die Namen sind selbsterklärend. Die Files in "Kurs" können durch Anklicken der aktionsfähigen Zellen (unbedingt der Reihe nach) abgearbeitet und dabei studiert werden.
• Les noms s'expliquent eux-mêmes. Les dossiers courrants peuvent être élaborés en marquant (par la souris) les cellules executables (maintenir absolument l'ordre) et en même temps isl peuvent être étudiés.

▪ **Anmerkung zum Titel "Kleine Einführung in Mathematica":**
€ Remarque quant au titre "Petite introduction en *Mathematica*":

Verglichen mit einem hohen Berg ist auch ein grosses Haus eine kleine Sache.
"Kleine Einführung" muss noch lange nicht "kurze Einführung" bedeuten ....
• En comparaison d'une haute montagne, même une grande maison n'est qu'une petite chose. "Petite introduction" ne signifie pas forcément "courte introduction" ...

■ VI. Literatur zu den Files
€ Littérature pour dossiers

S. Wolfram: *Mathematica*
Nancy Blachman: *Mathematica*: A Practical Approach
Nancy Blachman: *Mathematica* griffbereit

Rundgang in *Mathematica*
€ Tour en *Mathematica*

(Nach Ideen aus: Handbuch "Mathematica" von S. Wolfram)
(Selon des idées pris dans le manual "Mathematica" de S. Wolfram)
WIR94/98/99

1. Numerische Rechnungen
€ Calculs numériques

■ Taschenrechner
€ Calculatrice de poche

```
9 + 17
```

■ Exakte Resultate
€ Résulats exacts

```
3^101
```

■ Vorheriges Resultat approximiert
€ Résultat précédent approximé

```
N[%]
```

■ Numerische Resultate beliebiger Genauigkeit
€ Résultats numériques de précision quelconque

```
N[Sqrt[10],50]
```

■ Komplex rechnen
€ Calculs complexes

```
(4 + 7I)^12
```

■ **Mathematische Standardfunktionen**
€ Fonctions mathematiques standard

```
BesselJ[0,15.2]
```

■ **Nullstellen**
€ Des zéros

```
FindRoot[BesselJ[0,x],{x,14.5}]
```

■ **Beliebige Genauigkeit von Resultaten**
€ Précision quelconque de résultats

```
N[Zeta[1/2+13I],36]
```

■ **Numerische Integrale**
Intégraux numériques

```
NIntegrate[Sin[Sin[x]],{x,0,Pi}]
```

■ **Exakte Rechnungen mit natürlichen Zahlen**
€ Calculs exacts avec les nombres naturels

```
FactorInteger[70987635488]
```

■ **"Putzmaschine" einsetzen**
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

2. Graphiken
€ Graphiques

■ **Graph einer Funktion**
€ Graphe d'une fonction

```
Plot[Sin[Exp[x]],{x,0,Pi}]
```

```
Plot[Sin[Exp[x]],{x,0,Pi}];
```


Was war der Unterschied?
• Quel était la différence?

■ **Optionen zum Manipulieren von Plot**
€ Options pour manipuler Plot

```
Show[%,Frame->True,FrameLabel->{"Zeit","Weg"},
GridLines->Automatic];
```

■ **Höhenlinienkarte**
€ Carte des courbes à niveau

```
ContourPlot[Sin[x+Sin[y]],{x,-2,2},{y,-2,2}];
```

■ **3-dimensionale Graphen**
€ Graphes à 3 dimensions

```
Plot3D[Sin[x+Sin[y]],{x,-3,3},{y,-3,3}];
```

■ **3-dimensionale parametrisierte Flächen**
€ Surfaces paramétrisées à 3 dimensions

```
ParametricPlot3D[{u Sin[t], u Cos[t], t/3},
{t,0,15},{u,-1,1},Ticks->None];
```

■ **Oder vielleicht so**
€ Ou peut-être ainsi

```
ParametricPlot3D[{Sin[t], Sin[2t] Sin[u], Sin[2t] Cos[u]},
{t,-Pi/2,Pi/2},{u,0,2 Pi},Ticks->None];
```

■ **Die beiden letzten Graphen kombiniert**
€ Les deux derniers graphes combinés

```
Show[%,%%];
```

■ **Arbeit mit eingebauten Graphik-Komponenten**
€ Travail avec des composentes de graphiques incorporées

```
Show[Graphics3D[
{Cuboid[{0,0,0}],Cuboid[{2,2,2}],
 Cuboid[{2,1,3}],Cuboid[{3,2,1}],
 Cuboid[{2,1,1}]} ]]
```

■ **Musik**
€ Musique (Cela dure peut-être longtemps, s.v.p.

▪ **(Das hier dauert etwas lang, bitte warten)**
€ (Cela dure peut-être longtemps, s.v.p.)

```
Play[Sin[1000/t],{t,-4,16}]
```

■ **"Putzmaschine" einsetzen**
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

3. Algebra und Analysis
€ Algèbre et analyse

■ **Ausdrücke umformen**
€ Transformer des expressions

▪ **Ausdruck eingeben**
€ Entrer une expression

```
9(2+x)(x+y)+(x+y)^2
```

▪ **Ausdruck hoch 3 ausmultiplizieren**
€ Multiplier une expression puissance 3

```
Expand[%^3]
```


▪ **Ausdruck wieder faktorisieren**
€ Refactoriser une expression

```
Factor[%]
```

■ **Formal integrieren**
€ Intégrer formellement

```
Integrate[x^2 Sin[x]^2,x]
```

■ **Resultat wieder differenzieren**
€ Redifférencier le résultat

```
D[%,x]
```

■ **Resultat vereinfachen**
€ Simplifier le résultat

```
Simplify[%]
```

■ **Resultat als Funktion von x in eine Potenzreihe entwickeln**
€ Eévelopper le résultat comme fonction de x en une série de puissances

```
Series[%,{x,0,14}]
```

■ **Symbolisch gegebene Funktion in eine Potenzreihe entwickeln**
€ Développer le résultat comme fonction donnée symboliquement en une série de puissances

```
Clear[f];
Series[ (f[x + h]-f[x - h])/(2h),{h,0,6}]
```

■ **"Putzmaschine" einsetzen**
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

4. Gleichungen lösen
€ Résoudre des équations

■ Einfache Gleichung behandeln
€ Traiter des équations simples

▪ **Gleichung eingeben**
€ Entrer une équation

```
x^3 - 7 x^2 + 3 a x == 0
```

▪ **Gleichung nach x auflösen**
€ Résoudre une équation d'après x

```
Solve[%,x]
```

■ Gleichungssystem lösen nach x und y
€ Résoudre le système d'équation d'après x et y

```
Solve[{a x + b y == 0, x + y == c},{x,y}]
```

■ Numerische Lösung einer Gleichung höherer Ordnung
€ Solution numérique d'une équation d'ordre supérieur

```
NSolve[x^5+2x+1==0,x]
```

■ Transzendentes Gleichungssystem numerisch lösen
€ Résoudre numériquement un système d'équations transcendant

```
FindRoot[{Sin[x]==x-y,Cos[y]==x+y},{x,1},{y,0}]
```

■ Differentialgleichung lösen
€ Résoudre une équation différentielle

```
DSolve[y''[x]-k y[x]==1,y[x],x]
```


■ Differentialgleichung numerisch lösen und Graph ausgeben
€ Résoudre numériquement une équation différentielle et sortir le graphe

▪ **Gleichung lösen**
€ Résoudre équation

```
NDSolve[{y''[x] + Sin[x]^2 y'[x] + y[x] == Cos[x]^2,
        y[0]==1, y'[0]==0},
        y, {x,0,20}]
```

▪ **Graph ausgeben**
€ Sortir un graphe

```
Plot[Evaluate[y[x]/.%],{x,0,20}];
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

5. Listen
€ Listes

■ Beispiel: Liste machen und manipulieren
€ Exemple: Dresser une liste et la manipuler

▪ **Liste von Fakultäten**
€ Liste de factoriaux

```
Table[n!,{n,1,15}]
```

▪ **Logarithmus davon**
€ Logarithme

```
N[Log[%]]
```

- **Liste graphisch darstellen**
  € Faire une graphique de la liste

```
ListPlot[%];
```

- **Optimale Parabel durch letzten Graphen legen**
  € Poser une parabole optimale à travers de la liste

```
Fit[%%,{1,x,x^2},x]
```

```
Plot[%,{x,0,14}];
```

■ Beispiel: 2-dimensionales Array manipulieren
€ Exemple: manipuler un array bidimensionnel

- **Array kreieren: Teilerfremde Paare von andern unterscheiden, Zahlen nicht ausgeben (zu viele)**
  € Créer un array: Distinguer des paires sans facteur commun d'autres paires, ne pas sortir les nombres
  (il y en a trop)

```
arr = Table[If[GCD[i,j]==1,1,0],{i,30},{j,30}];
```

- **Dichte-Graph**
  € Graphique de la densité

```
ListDensityPlot[arr]
```

- **Grösster absoluter Wert in der Fourier-Transformierten von `arr`**
  € Plus grande valeur absolue dans la transformée de Fourier de `arr`

```
??Fourier
```

```
Fourier[{1, 1, 1, 1, -1, -1, -1, -1}]
```

```
Max[Abs[Fourier[{1, 1, 1, 1, -1, -1, -1, -1}]]]
```

```
Fourier[{1, 1, 1, 1, -1, -1, -1, -1}] // N
```

```
Max[Abs[N[Fourier[{1, 1, 1, 1, -1, -1, -1, -1}]]]]
```

```
Max[Abs[N[Fourier[arr]]]]
```

Frage: Was ist von der Ausgabe des Resultats zu halten?
• Question: Que faut-il penser de la sortie du résultat?

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

6. Matrizen
€ Matrices

■ Beispiel: Matrix eingeben und damit rechnen
€ Exemple: Entrer une matrice et calculer avec elle

- **Matrix generieren**
  € Générer une matrice

```
m = Table[1/(i+j+1),{i,3},{j,3}]
```

- **Anschauen**
  € Considérer

```
MatrixForm[m]
```

- **Inverse von `m`**
  € Inverse de `m`

```
Inverse[m]
```

- **Anschauen**
  € Considérer

```
MatrixForm[Inverse[m]]
```

- **Matrix mit Inverser multiplizieren, anschauen (Einheitsmatrix)**
  € Multiplier la matrice avec l'inverse, considérer
  (matrice unité - unifiée)

```
MatrixForm[m.Inverse[m]]
```

- **Neue Matrix mit m kreieren**
  € Créer une nouvelle matrice avec m

```
mNeu = m - x IdentityMatrix[3]
```

- **Anschauen**
  € Considérer

```
MatrixForm[mNeu]
```

- **Determinante von mNeu**
  € Déterminante de mNeu

```
Det[mNeu]
```

- **Eigenwerte von m**
  € Valeurs propres de m

```
Eigenvalues[m]
```

- **Eigenwerte numerisch**
  € Valeurs propres numériques

```
N[%] // Chop
```

■ Beispiel: Rechnen mit einer grossen Matrix
€ Exemple: Calculer avec une grande matrice

- **Zufällige 100 x 100-Matrix kreieren, nicht ausgeben**
  € Créer une matrice 100 x 100 fortuite, ne pas la sortir

```
n = Table[Random[],{100},{100}];
```

- **Davon Eigenwerte (komplex), nicht ausgeben**
  € En calculer les valeurs propres (complexes), ne pas les sortir

```
Eigenvalues[%];
```


- **Eigenwerte als Punkte in der komplexen Ebene graphisch darstellen**
  € Représenter graphiquement les valeurs propres comme points dans le plan complex

```
ListPlot[Transpose[{Re[%],Im[%]}]]
```

■ Beispiel: Rechnen mit symbolischen Matrizen
€ Exemple: Calculer avec des matrices symboliques

- **Matrix geben**
  € Donner une matrice

```
s = {{a,b},{-b,2a}}; MatrixForm[s]
```

- **Davon Eigenvektoren, algebraisch vereinfacht**
  € En donner les vecteurs propres, simplifiés algébraiquement

```
Simplify[Eigenvectors[s]]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

7. Transformationsregeln und Definitionen
€ Règles de transformation et définitions

■ Ersetzungsregeln
€ Règles de remplacement

- **Beispiel: Regel zur Ersetzung von x in einem Ausdruck y durch 1+a**
  € Exemple: Règle pour remplacer x dans une expression y par 1 + a

```
y = 1 + x^2 + 3 x^3
```

```
y /. x->1+a
```

- **Beispiel: In einem Ausdruck f[2] ersetzen durch b**
  € Exemple: Dans une expression remplacer f[2] par b

```
{f[1],f[2],f[3]} /. f[2]->b
```

- **Beispiel: In einem Ausdruck f[n] ersetzen durch n^2 (n^2)**
  € Exemple: Dans une expression remplacer f[n] par n^2 (n^2)

```
{f[1],f[2],f[3]} /. f[n_]->n^2
```

■ Definition einer Funktion
€ Définition d'une fonction

- **Definition : f[n] sei n^2(n^2)**
  € Définition : f[n] soit n^2(n^2)

```
f[n_]:= n^2;
(*Ausgabe auf dem Schirm: € Imprimer sur l'écran: *)
f[n]
```

- **Dann rechnen mit Funktionswerten**
  € Calculer ensuite avec des valeurs fonctionelles

```
f[3] + f[a+b]
```

■ Rekursive Definition einer Funktion (Fakultäten)
€ Définition récursive d'une fonction (factoriaux)

- **Definition**
  € Définition

```
fac[n_]:= n fac[n-1]
```

- **Initialisation**
  € Initialisation

```
fac[1]:= 1
```


- **Abfrage**
  € Interrogation

```
?fac
```

- **Anwendung: Berechnung von 25!**
  € Application : Calcul de 25!

```
fac[25]
```

■ Bsp.: Eingabe der Rechenregeln für den Logarithmus
€ Ex.: Entrer des règles de calcul pour le logarithme

- **Definition**
  € Définition

```
log[x_ y_]:= log[x] + log[y]
```

- **Anwendung: Berechnung von log[a b c d]**
  € Application: Calcul de log[a b c d]

```
log[a b c d]
```

■ Bsp.: Ausschaltung der Regel, dass bei einer Definition links vom Gleichheitszeichen kein Operationszeichen zwischen Funktionen stehen darf
€ Bsp.: Elimination de la règle que dans une définition à gauche du signe égal il est interdit de placer un signe d'opération entre des fonctions

- **Versuch einer Definition**
  € Essai de définition

```
g[i_] + g[j_] := g[i+j]
```

- **Richtige Definition so:**
  € Définition correcte ainsi;

```
g/: g[i_] + g[j_] := g[i+j]
```

- **Anwendung auf eine Berechnung**
  € Application pour un calcul

```
Clear[y];
g[x] + g[y] + g[z]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

8. Symbolisches Rechnen

■ Manipulation einer Liste
€ Manipulation d'une liste

- **Permutationen**
  € Permutations

```
Permutations[{a,b,c}]
```

- **Aufhebung der Untergruppen**
  Abolir les sous-groupes

```
Flatten[%]
```

- **Position von b angeben**
  € Donner la position de b

```
Position[%,b]
```

- **Letzte Liste aufmultiplizieren**
  € Multiplier la dernière liste

```
FoldList[Times,{1},%]
```


■ Geschachtelte Listen
€ Listes emboîtées

- **Definition der Liste**
  € Définition de la liste

```
NestList[Cos,x,3]
```

- **Graph dieser Funktionen**
  € Graphiques de ces fonctions

```
Plot[Evaluate[%],{x,0,1}];
```

■ Anwenden einer Funktion simultan auf verschiedene Elemente
€ Applications d'une fonction simultanément à différents éléments

- **Beispiel**
  € Exemple

```
Clear[f];
Map[f,{a,b,c,d}]
```

- **Beispiel mit namenlosen Funktionen (mit "Function")**
  € Exemple avec des fonctions sans nom (avec "Function")

```
Map[Function[x,1+x^2],{a,b,c,d}]
```

■ Bsp.: Rekursive Berechnung der Fibonacci-Folge, Verfolgung des Rechenwegs
€ Bsp.: Calcul récursif de la suite de Fibonacci, poursuite de la voie de calcul

- **Definition**
  € Définition

```
Clear[f];
f[0]=f[1]=1;
f[n_]:=f[n-1]+f[n-2]
```

- **Anwendung: Berechnung f[4]**
  € Application: Calcul de f[4]

```
f[4]
```

- **Rechenweg verfolgen**
  € Poursuivre la voie de calcul

```
Trace[f[4],f[_]]
```

- **"Putzmaschine" einsetzen**
  € Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

9. Programmierung
€ Programmation

- **Programm zur Erzeugung einer nxn-Hilbert-Matrix**
  € Programme pour la création d'une matrice n x n de Hilbert

- **Programm**
  € Programme

```
Hilbert[n_]:=Table[1/(i+j-1),{i,n},{j,n}]
```

- **Aufruf für eine 3x3-Matrix**
  € Appel pour une matrice 3 x 3

```
Hilbert[3]; MatrixForm[%]
```


- **Programm zur Erzeugung des charakteristischen Polynoms einer Matrix**
  € Programme pour la création du polynome caractéristique d'une matrice

- **Programm**
  € Programme

```
CharPoly[m_,x_]:=
Det[m-x IdentityMatrix[Length[m]]]/; MatrixQ[m]
 (* Check auf Matrix bei m *)
```

- **Wieso /; ?**
  € Pourquoi /; ?

```
?/;
```

- **Anwendung**
  € Application

```
m = {{1,2,3},{2,3,4},{3,4,5}};
Print[MatrixForm[m]];
CharPoly[m,x]

m = {1,2,3,2,3,4,3,4,5};
Print[MatrixForm[m]];
CharPoly[m,x]
```

- **Programm zur Auffindung der nächsten Primzahl**
  € Programme pour trouver le prochain nombre premier

- **Programm**
  € Programme

```
NextPrime[n_Integer]:=
Module[{k=n},
        While[!PrimeQ[k],k++];
        Return[k]
      ]
```

- **Anwendung auf n =111**
  € Application pour n = 111

```
NextPrime[111]
```

■ Programm zur Berechnung statistischer Grössen, ohne Loops zu verwenden (elegant)
€ Programme pour calculer des valeurs statistiques, sans utiliser des loops (élégant)

- **Programm**
  € Programme

```
Mean[list_List]    :=Apply[Plus,list] / Length[list];

Variance[list_List]:=Mean[(list-Mean[list])^2];

Quantile[list_List,q_]:=
   Part[Sort[list],-Floor[-q Length[list]]]/; 0<q<1;

Alles[list_, q_]:=Module[{},
       Print["Mean = ",Mean[list]];
       Print["Variance = ",Variance[list]];
       Print["Quantile = ",Quantile[list,q]];
                      ]

??Module
```

- **Anwendung**
  € Application

```
l={1,2,2,3,3,3,4,4,4,5,5,6,7,7,8,9}; q=0.1;
Alles[l,q]
```

■ Programm "Zufallsspaziergang" - was macht dieses Programm?
€ Programme "promenade au hasard" - que fait cd programme?

- **Programm**
  € Programme

```
zs[n_Integer]:=
   FoldList[Plus,0,Table[Random[]-1/2,{n}]]
```


- **Anwendung**
  € Application

```
zs[100]
```

■ Programm, das die ersten n Terme in der Kettenbruchentwicklung einer Zahl findet: elegante funktionale Programmierung
€ Programme qui trouve les n premiers termes d'un nombre

- **Programm**
  € Programme

```
ContinuedFraction[x_Real,n_Integer]:=
      Floor[NestList[Function[{u},1/(u-Floor[u])],
                     x,n-1]]
```

- **Anwendung: Goldener Schnitt in Kettenbruchentwicklung**
  € Application: Section d'or en évolution de fractions continues

```
h=N[(Sqrt[5]-1)/2,100];
Print[h]
ContinuedFraction[h,30]
```

- **Anwendung: e in Kettenbruchentwicklung**
  € Application: e en évolution de fractions continues

```
e=N[E,100];
Print[e]
ContinuedFraction[e,30]
```

- **Anwendung: Pi in Kettenbruchentwicklung**
  € Application: Pi en évolution de fractions continues

```
p=N[Pi,100];
Print[p]
ContinuedFraction[p,30]
```

■ **Programm mit Mix aus symbolischer Konstruktion und mathematischen Operationen: Ableiten der Funktionen einer Liste nach allen Variablen einer andern Liste**
€ Programme avec mélange de construction symbolique et opérations mathématiques: Dériver les fonctions d'une liste d'après toutes les variables d'une autre liste

▪ **Programm**
€ Programme

```
Jac[funs_List,vars_List]:=Outer[D, funs, vars]
```

▪ **Anwendung**
€ Application

```
f={x^2+x y-1, Cos[y]-z, a x z};
v={x,y,z};
Jac[f,v]
```

■ **Liste von Regeln für Laplace-Transformationen**
€ Liste de règles pour des transformations de Laplace

▪ **Programm (Regeln)**
€ Programme (règles)

```
Laplace[c_,t_,s_]:=c/s/; FreeQ[c,t];
Laplace[a_+b_,t_,s_]:=
            Laplace[a,t,s]+Laplace[b,t,s];
Laplace[c_ a_,t_,s_]:=
            c Laplace[a,t,s]/; FreeQ[c,t];
Laplace[t_^n_.,t_,s_]:=
            n!/s^(n+1)/; (FreeQ[n,t]&&n>0);
Laplace[a_. Exp[b_.+c_. t_],t_,s_]:=
       Laplace[a Exp[b],t,s-c]/; FreeQ[{b,c},t];

?_.

?FreeQ
```


▪ **Anwendung**
€ Application

```
Laplace[E^t+4t^2-2t+1,t,s]
```

■ **Programm, das pattern matching benutzt um die Anzahl gleicher Elemente einer Liste auszuzählen - kurz, elegant, effizient!**
€ Programme qui utilise le "pattern matching" pour compter le nombre d'éléments égaux d'une liste - bréf, élégant, efficase!

▪ **Programm**
€ Programme

```
RunEncode[{rest___Integer,same:(n_Integer)..}]:=
    Append[RunEncode[{rest}],{n,Length[{same}]}];
RunEncode[{}]:={}
```

▪ **Anwendung**
€ Application

```
r={1,2,2,2,3,3,3,3,3,3,3,4,4,5,6,6,6,7,8,9,9,9,10};
RunEncode[r]
```

■ **Programm, um Graphik zu erzeugen**
€ Programme pour créer des graphiques

▪ **Programm**
€ Programme

```
PolarPlot[r_,{t_,tmin_,tmax_}]:=
    ParametricPlot[{r Cos[t], r Sin[t]},
         {t,tmin,tmax},AspectRatio->Automatic]
```

▪ **Anwendungen**
€ Applications

```
PolarPlot[t^2,{t,0,2 Pi}];

PolarPlot[t^(1/2),{t,0,2 Pi}];

PolarPlot[E^t,{t,0,Pi}];
```

■ Programm, das die Lösungen einer Polynom-Gleichung als Punkte der komplexen Ebene darstellt
€ Programme qui représente les solutions d'une équation polynomiale comme points d'un plan complex

- **Programm**
  € Programme

```
RootPlot[poly_,z_]:=
   ListPlot[{Re[z],Im[z]}/. NSolve[poly==0,z],
   AspectRatio->Automatic,
        PlotStyle->{PointSize[0.04]}]/;
                   PolynomialQ[poly,z]
```

- **Anwendung: Einheitswurzeln**
  € Application: Racines unifées (de l'unité)

```
RootPlot[z^20 -1,z];
```

■ Programm, das Zahlen in Matrixform aus einem File liest und als 3D-Graphik darstellt
€ Programme qui lit des nombres en forme de matrice dans un dossier et les représente en graphique 3D

- **Programm**
  € Programme

```
file[file_String]:=
    ReadList[file,Number,RecordLists->True]

FilePlot3D[file_String]:=
    ListPlot3D[ReadList[file,Number,
                        RecordLists->True]]
```

- **Anwendung**
  € Application

```
Print[file["f:\mate\Daten\aadaten"]];
FilePlot3D["f:\mate\daten\aadaten"]
```


- **Anhang**
  € Annexe

```
?RecordLists

!!f:\Mathe/Daten/aadaten

<<f:\Mathe/Daten/aadaten
```

Was ist jetzt passiert?

```
??<<
```

■ Programm, das externe Daten manipuliert - hier: das Files sucht mit gegebenen Teilstrings
€ Programme qui manipule des données externes - Ici: le dossier cherche par des stings partiels donnés

- **Programm**
  € Programme

```
?Select

fn=FileNames[] >> f:\Mathe/Daten/aaxxx;
fnxxx=Flatten[ReadList["f:\Mathe/Daten/aaxxx",
                Expression,RecordLists->True,
                WordSeparators->{" "}]];
Print[fnxxx];
Print[fnxxx[[1]]];
Print[fnxxx[[3]]];
Clear[Where];
Where[s_String]:=
   Select[fnxxx,(Length[FindList[#,s,1]]>0)&]
```

- **Anwendung: Suche nach *e* im momentanen directory**
  € Application: Cherche *e* dans le directoire momentané

```
Where["e"]
```

Wo liegt also das Problem hier? • Où est le problème ici?

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

10. *Mathematica* Packages
€ *Mathematica* Packages

■ Beispiel: Laplace-Transformationen
€ Exemple: Transformations de Laplace

- **Package laden**
  € Charger le Package

```
<<Calculus`LaplaceTransform`
```

- **Eine Transformierte rechnen**
  € Calculer une transformée

```
LaplaceTransform[t^n Exp[-c/t],t,s]
```

■ Beispiel: Padé-Approximation
€ Exemple: Approximation de Padé

- **Package laden (neuer Kern benutzen!)**
  Charger le Package (employer un nouveau kernel!)

```
<<Calculus`Pade`
```

- **Eine Approximation rechnen**
  € Calculer une approximation

```
Pade[Exp[Sin[x]],{x, 0, 2, 3}]
```

■ Beispiel aus der Statistik
€ Exemple de la statistique

- **Package laden**
  € Charger le Package

```
<<Statistics`DescriptiveStatistics`
```


- **Einige Kenngrössen rechnen**
  € Calculer quelques valeurs spécifiques

```
LocationReport[Table[Random[],{1000}]]
```

■ Beispiel: Permutationen
€ Exemple: Permutations

- **Package laden**
  € Charger le Package

```
<<DiscreteMath`Permutations`
```

- **Zufällige Permutation erzeugen**
  € Créer une permutation par le hazard

```
RandomPermutation[20]
```

- **Zyklische Zerlegung**
  € Décomposition cyclique

```
ToCycles[%]
```

■ Beispiel aus der Chemie
€ Exemple pris de la chemie

- **Package laden**
  € Charger le Package

```
<<Miscellaneous`ChemicalElements`
```

- **Ein Atomgewicht abrufen**
  € Rappeler un poids atomique

```
AtomicWeight[Tungsten]
```

- **Noch eines**
  € Encore un

```
AtomicWeight[Plutonium]
```

- **Ausschalten der Ausgabe "unstabil"**
  € Eliminer la sortie "instable"

```
Off[AtomicWeight::unstable]
```

- **Atomnummer**
  € Nombre atomique

```
AtomicNumber[Plutonium]
```

- **Verhältnis Gewicht zu Nummer**
  € Correlation entre poids et nombre

```
ListPlot[AtomicWeight[Elements]/AtomicNumber[
        Elements], PlotJoined->True]
```

- **Beispiel: Sternform eines Platonischen Körpers**
  € Exemple: Forme d'étoile d'un corps platonique

- **Package laden**
  € Charger le Package

```
<<Graphics`Polyhedra`
```

- **Sternform des Ikosaeders**
  € Forme d'étoile de l'icosaèdre

```
Show[Polyhedron[GreatIcosahedron]];
```

- **"Putzmaschine" einsetzen**
  € Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

11. Datenaustausch mit *Mathematica*, andere Ausgabeformen
Achtung: Einige Befehle sind für UNIx, andere für DOS, andere für Windows...

€ Echange de données par *Mathematica,* autres formes de sorties
Attention: Quelques ordres sont pour UNIX, d'autres pour DOS, d'autres pour Windows...

- **Beispiel Ausgabeformen**
  € Exemple de formes de sorties

- **Beispiel eines betrachteten Ausdrucks, Standardausgabe**
  € Exemple d'une expression considérée, output standardisé

```
Remove["Global`@*"]
```

```
c=(a^2+b^2)/(x+y)^5
```

$$\frac{a^2 + b^2}{(x + y)^5}$$

- **Ausgabe in der Eingabeform**
  € Output en forme d'entrée

```
InputForm[c]
```

- **Ausgabe in TeX-Form (Profi-Satzsystem)**
  € Output en forme TeX (système de composition professionel)

```
TeXForm[c]
```

- **Ausgabe in Fortran-Form**
  € Output en forme Fortran

```
FortranForm[c]
```

- **Ausgabe in C-Form**
  € Output en forme C

```
CForm[c]
```

■ Beispiel: Kommunikation mit der Umgebung
€ Exemple: Communication avec l'entourage

■ **Externe Daten lesen**
(file "C:\WORK\aaadaten.txt")
€ Lire de données externes (dossiers)

```
$Path

$Path = Join[$Path, {"C:\\WORK"}]

?? $Path

!cd C:\\WORK;

! cd C: \work;
ReadList["!dir", String]

ReadList["AAAdaten.txt", String]

!cd C:\work;
ReadList["C:\work\AAAdaten.txt",String]

ReadList["AAAdaten.txt", Number]

! cd C: \\ work;
ReadList["C:\\work\\AAAdaten.txt", String]

ReadList["C:\\work\\AAAdaten.txt", String]

ReadList["C:\work\\(\*ErrorBox[\(\*StyleBox[A daten]\)]\).txt", String,
 RecordLists -> True]

! cd C: \work;
ReadList["C:\work\Adaten", String]

FileNames["c:\\work\\
   \(\*ErrorBox[\(\*StyleBox[", Evaluatable -> False, AspectRatioFixed -> True,
         FontFamily -> "Courier", FontSize -> 12]\)]\)]

!cd

data = "a b c d e f 1 2 3 4 5 6 1.1 1.2. 1.3 aa ab ac"

data >> c:\work\ABdata

ReadList["c:\work\ABdata", String]
```

■ **Externer Befehl wie "square" etc. ausführen und dann Daten einlesen (Funktioniert nicht mit "square" , da Befehl nicht vorhanden)**

€ Executer un ordre externe tel que "square" etc. et lire les données ensuite (ne fonctionne pas avec "square", car l'ordre n'est pas disponible

Die restlichen Befehle sind UNIX-Befehle. "cd": Directory-Wechsel, "ls": Directory lesen.
• Les autres ordres sont des ordres UNIX. "cd": Changer le directoire, "ls": Lire le directoire

```
ReadList["!square",Number]

ReadList["!ls",String]

ReadList["!cd C:/work;ls",String]

!cd work;
ReadList["!dir",String]

ReadList["!cd C:\work\AAAdaten.txt;dir",String]

!cd
```

■ **Externer Befehl "iterat" in *Mathematica* einbinden**
(Funktioniert nicht, da Befehl nicht vorhanden)
€ Relir l'ordre externe "iterat" à *Mathematica*
(Ne fonctionne pas parce que l'ordre n'est pas présent)

```
Install["iterat"]

getdata["C:\work\AAAdaten.txt",8]
```

■ **Nach String "1" im File "AAAdaten" suchen**
€ Cherger d'après String "l" lans le dossier "AAAdaten"

```
FindList["C:\work\AAAdaten.txt","1"]
```

■ **String ersetzen**
€ Remplacer String

```
StringReplace[%,"7"->"a"]
```

■ **Suchen in externen Verzeichnissen**
€ Cherger dans des listes externes

```
FindList["C:\work\*","AAA"]

FindList["!cd C:\work;ls","A"]

FindList["!cd C:\work;ls","AA"]

FindList["!cd C:\work;ls","AAA"]
```

```
FindList["!cd c:\work;ls","AAA_"]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

1. Einstieg in *Mathematica*
€ Premier contact avec *Mathematica*

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 1 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 1.
WIR94/99

■ 1.1. Maschinenspezifisches
€ Problèmes spécifiques à la machine

■ Aufgabe € Problème

Finde heraus, was von *Mathematica* und was von der Maschine abhängt!
• Distingue ce qui dépend *Mathematica* et ce qui dépend de la machine.

Hinweis: Wenn Du einmal nicht weiterkommst, so frage den Kursleiter oder konsultiere die Literatur, z.B. "***Mathematica***" von Wolfram.
• **Indication:** Si tu as un problème, adresse-toi au professeur ou consulte la littérature, p.ex. "***Mathematica***" de Wolfram.

■ 1.2. Versionspezifisches
€ Problèmes spécifiques de la version

Mit welcher *Mathematica*-Version arbeitest Du? Drücke die Enter-Taste in der Zelle mit "$Version".
• Avec quelle version de *Mathematica* traveilles-tu? Appuie sur la touche "entrer" dans la cellule avec "$Version".


```
$Version
```

■ 1.3. Front-End versus Kernel
€ Front-End versus Kernel

■ Aufgabe: € Problème:

1. Finde heraus, was "Front-End" und was "Kernel" bedeutet.
• Que siginifie "Front-End" et "Kernel"?
2. Was bewirkt die Enter-Taste?
• Que produit la touche "Enter"?
3. Was bewirkt "Shift-Return"?
• Que produit la touche ""Shift-Return"?
4. Was bewirkt "Command"?
• Que produit la touche "Command"?
5. Was bewirken die Icons "Action" - "Interrupt" etc.?
• Que produisent les Icons "Action" - "Interrupt" etc.?
6. Wie ist es auf andern Maschinen?
• Comment cela se passe-t-il sur d'autres machines?

■ 1.4. Notation
€ Notation

■ Aufgabe € Problème

■ Probiere aus: € Essaie:

```
Apply[Plus, {a,b,c,d}]
```

```
a + b + c + d
```

Wie ist es mit Gross- und Kleinschreibung?
Wie ist es mit den verschiedenen Klammerarten?
Wie ist es mit den "blanks"?
Wie ist es mit der Interpunktion?
Wie ist es mit dem Fond? (Fixe und variable Abstände, z. B. Courier contra Times)?
•
Pose-toi des Questions aux sujets suivants:
Majuscule - minuscule,
parenthèses,
"blanks",
ponctuation,
"fond" (distances foxes et variables, p. ex. Courier contra Times).

■ 1.5. *Mathematica* interaktiv benutzen: € Utiliser la machine interactivement:

■ Aufgabe € Problème

■ Probiere aus: € Essaie:

Potenzieren: • Calculer les puissances:

```
5^10
```

Resultat der letzten Zelle verwenden: • Utiliser le résultat de la dernière cellule:

```
%^(1/10)
```

```
% + a
```

```
%3^(1/10)
```

```
%%% + b
```

Namen vergeben: • Donner des noms:

```
Schifflänge = 3
```

```
Schifflänge
```

```
20 Schifflänge
```

```
2 m + 3 m
```

■ 1.6. Reserviere Namen: € Réserve des noms:

Mathematica hat sehr viele Befehle (Funktionen etc.) eingebaut.
Merke: *Mathematica*-Namen beginnen alle mit Grossbuchstaben.
• Dans *Mathematica* beaucoup d'ordres (fonctions etc.) sont incorporés.
Retenez: Les noms de *Mathematica* commencent tous par des majuscules.

■ Aufgabe € Problème:

■ Probiere aus: € Essaie:

```
Abs[-8]
```

```
Cos[Pi]
```

```
D[x^2-3x+4,x]
```



```
M = {{1,2},{5,1}}; MatrixForm[M]
```

```
Det[M]
```

```
GCD[48005, 100098630]
```

■ 1.7. Help-Funktion und Kommando-Vervollständigung: € Fonction Help et complètement du commandement:

Mathematica hat mehr als 700 Funktionen eingebaut, die über das Help-Service abfragbar sind. "*" figuriert dabei als "wild card". Help ist Front End-spezifisch.
Weitere Info: *Mathematica* Quick Reference Guide, *Mathematica* Help Stack, Manual, spez. Literatur.
• Dans *Mathematica* sont incorporées plus de 700 fonctions qui sont disponibles par le service Help. "*" y figure comme "wild card". Help est spécifique de Front End. Autres Informations: Quick Reference Guide, *Mathematica* Help Stack, Manual, litt. spéc.. Literatur.

■ Aufgabe € Problème

■ Probiere aus: € Essaie:

```
?Factor*
```

```
?FactorInteger
```

```
FactorInteger[75]
```

```
FactorInteger[1122445667788]
```

```
?Sin
```

```
?*Sin*
```

```
?Cos
```

```
?Tan
```

```
?Exp*
```

```
?Log
```

```
?Divisors
```

```
Divisors[5000]
```

```
?*Graphic*
```

```
?*Plot*
```

```
??Plot
```

```
?@
```

■ **Rufe die Information ab zu folgenden Ausdrücken:**
€ Cherche l'information pour les expressions suivantes:

Beispiel: • Exemple:

```
?NIntegrate
```

Integrate
ContourPlot
InterpolatingPolinomial
MapAt
Binomial
Eigenvalues
FindRoot
Integrate
Timing

■ **Kommando vervollständigen**
€ Compléter le commandement

■ **Probiere aus: € Essaie:**

Gehe in die nächste Zelle und drücke "Command k":
• Entrer dans la cellule suivante et appuyer "Command k" :

```
Sor
```

Gehe in die nächste Zelle, markiere mit der Maus "So" und drücke "Command k":
• Entrer dans lan cellule suivante, marquer à la sourie "So" et presser "Command k":

```
Sor
```

Probiere aus:
• Essayer:

```
Sort[{2,5,3,1,8}]
```

Gehe ins Menu auf ("Action" (old),) "Prepare Input", "Complete Selection":
• Entrer dans le menu sur ("Action" (old),) "Prepare Input", "Complete Selection":

```
Sor
```

Gehe ins Menu auf ("Action" (old),) "Prepare Input", "Make Template":
• Entrer dans le menu sur ("Action" (old),) "Prepare Input", "Make Template":

```
Sor
```

Gehe in die nächste Zelle und drücke "Command I": (Old?)
• Entrer dans la cellule suivante et presser "Command I": (Old?)


```
Sor
```

Gehe in die nächste Zelle und drücke "Command I": (Old)
Markiere "ContourPlot" in der nächste Zelle und drücke "F1".
Markiere die nächste Zelle und drücke "Shift F1".
•
Entrer dans la cellule suivante et appuier "Command I": (Old)
Marque "ContourPlot" dans la cellule suivante et appuie "F1".
Marque la cellule suivante et appuie "Shift F1".

```
ContourPlot
```

■ 1.8. Help-Funktion und Kommando-Vervollständigung bei Zeichen:
€ Fonction Help et complètement de commandement:

Was bedeuten die folgenden Zeichen? • Que signifient les signes suivants?
`+ - * / ^ ! < <= > >=`

```
?+
```

```
?-
```

```
?*
```

```
?/
```

```
?^
```

```
?!
```

```
?<
```

```
?<=
```

```
?>
```

```
?>=
```

■ **Probiere aus: € Essaie:**

```
4 + 7
```

```
4/2
```

```
4/3
```

```
N[4/3]
```

```
3 < 6
```

```
?<
```

```
Alias["<"]
```

```
Alias["="]
```

■ **Aufgabe € Problème:**

Wende "Alias" auf weitere mathematische Symbole an.
• Utilise "Alias" pour d'autres symboles mathématiques.

■ 1.9. Die verschiedenen Klammertypen € Les différents types de parenthèses

■ **Aufgabe € Problème**

Beachte im Folgenden, sofern vorhanden, die Klammern. Was passiert jeweils?
• Observe dans ce qui suit les parenthèses, quand il y en a.
Que se passe-t-il chaque fois?

■ **Probiere aus: € Essaie:**

```
1 + 2 + 3
```

```
1 + (2 + 3)
```

```
1 / 2 - 5
```

```
(1 + 2 ) * 3
```

```
(1 + 2 ) 3
```

```
1 / (2 - 5)
```

```
?Divisors
```

Argumente von Funktionen: • Arguments de fonctions

```
Divisors[100]
```

```
?Random
```

```
Random[ ]
```

```
Random[Integer]
```

```
Random[Integer,{50,60}]
```

Mengen, Listen, Vektoren, Matrizen...:
• Ensembles, listes, vecteurs, matrices...:

```
{x, x^2, x^3}
```

```
{{a[1,1],a[1,2]},{a[2,1],a[2,2]}}
```

```
MatrixForm[{{a[1,1],a[1,2]},{a[2,1],a[2,2]}}]
```



```
v={a,b,c}
```

```
v[[1]]
```

```
v[[2]]
```

```
m = {{1,2,3},{4,5,6}}
```

```
MatrixForm[m]
```

```
m[[2]]
```

```
m[[2,1]]
```

■ **Kommentare: € Commentaires:**

■ **Probiere aus: € Essaie:**

```
7 + 4 (* Das ist ein Kommentar *)
```

■ 1.10. Packages € Packages

Gewisse Befehle sind ausgelagert in sogenannten "Packages". (Vgl. in Menu: "Info", Open Function Browser" (old?) --> Help --_ Help.) Packages muss man erst laden, bevor man sie verwenden kann. (NeXT:" ` " ist Alt-Shift-n, vgl. Preferences.)
• Certains ordres sont stockés à l'extérieur dans les "Packages". (Compare dans le Menu: "Info", Open Function Browser" (old?) --> Help --_ Help.) Il faut d'abort charger Packages, avant de pouvoir les utiliser. (NeXT:" ` " est Alt-Shift-n, voir Preferences.)

■ **Probiere aus € Essaie:**

```
?Mean
```

```
Mean[list] gives the mean of the entries in list.
```

(Durch den Aufruf ist Mean jetzt definiert. • Par l'appel Mean est défini maintenant.)

```
Remove[Mean]
```

(Damit Mean neu definiert werden kann, muss das alte erst gelöscht werden. • Pour redéfinir Mean, il faut d'abort effacer la vieille version)

```
<<Statistics`DescriptiveStatistics`
```

```
?Mean
```

```
Mean[{1,2,3,4,5,6}]
```

Eine andere Art, ein Package zu laden:
• Une autre manière de charger un Package:

```
Needs["Graphics`Graphics`"]
```

```
PieChart[{
        {28,"Ingenieure"},
        {21,"CMS"},
        {20,"Wissensch."},
        {12,"Math."},
        {6,"Phys."},
        {15,2"Andere"}
    }];
```

■ Aufgabe € Problème

Versuche Dich selbst im Laden von Packages mit Hilfe von Menu: ("Info", Open Function Browser" (old)) --> Help--Help.
• Essaie toi-même à charger des Packages à l'aide du menu: ("Info", Open Function Browser" (old)) --> Help--Help.

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

1. Einstieg in *Mathematica*

€ Premier contact avec *Mathematica*

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach.... Hinweis: Kapitel 1 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 1.
WIR94/99

■ Aufgabe 1 € Problème 1

■ (a)

Finde alle Kommandos, die mit O beginnen ("Olga").
• Trouve tous les commandements qui commencent par O ("Olga").


```
?O*
```

■ (b)

Finde alle Kommandos, die das Teilwort "L i s t" enthalten.
• Trouve tous les commandements qui contiennent le mot (partie de mot) " L i s t".

```
?*List*
```

■ Aufgabe 2 € Problème 2

Teste alle Funktionen Deines Front-Ends.
• Examine tous les fonctions de ton Front-End

■ Aufgabe 3 € Problème 3

Verwende "Alias[ ]", um die eingebaute Liste der Entsprechungen zu den Abkürzungen zu generieren.
• Utilise "Alias[ ]", por générer la liste incorporée contenant les correspondances des abréviations.

```
Alias[ ]
```

■ Aufgabe 4 € Problème 4

Werte die folgenden Ausdrücke aus:
• Evalue les expressions suivantes:

```
3 x + 7 x
```

```
2 Meter + 13 Meter
```

```
120 lbs (1 kg/(2.2 lbs))
```

■ Aufgabe 5 € Problème 5

Lade ("Graphics.m" (old)) "Graphics.nb" und mache ein Balkendiagramm, das *Mathematica*-Benützer auf Jobs aufgeteilt zeigt.
• Charge ("Graphics.m" (old)) "Graphics.nb" et fais un diagramme à barres qui divise les utilisateurs de *Mathematica* selon leurs emplois.

```
<< Graphics`Graphics`;
BarChart[{{31,"Research"},
          {26,"Prof"},
          {15,"Eng"},
          {8,"Student"},
          {13,"CS"},
          {7,"Other"}}]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

2. Numerische Probleme
€ Problèmes numériques

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 2 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 2.
WIR94/99

Mathematica kennt ganze Zahlen (Integers), Fliesskommazahlen (mit Dezimalpunkt), rationale Zahlen, komplexe Zahlen sowie Zahlen, die durch Symbole gegeben sind (z.B. Pi).
• *Mathematica* connaît les nombres entiers (integers), les nombres avec virgule flottante (avec point décimal), les nombres rationaux, les nombres complexes ainsi les nombres représentés par des symboles.

■ 2.1. Arithmetische Operationen
€ Opérations arithmétiques

Probiere aus: • Essaie:

```
53 + 78
```

Multiplikation: • Multiplication:

```
127*9721
```

```
127 9721
```

Potenzieren: • Calculer les puissances:

```
34^56
```

```
%^(1/56)
```


■ 2.2. Rationale Zahlen
€ Nombres rationaux

■ ***Mathematica*** **rechnet exakt, wenn nicht anders verlangt!**
€ ***Mathematica*** **calcule exactement, sauf si on le veut autrement!**

Probiere aus: • Essaie:

```
2/4 + 24/144
```

```
2 + 2/5
```

■ 2.3. Irrationale Zahlen
€ Nombres irrationaux

■ ***Mathematica*** **rechnet exakt**
€ ***Mathematica*** **calcule exactement**

Beispiel: Quadratwurzel • Exemple: racine carrée

```
Sqrt[17]
```

```
Sqrt[17.0]
```

```
Sqrt[17]
```

```
Sqrt[17]//N
```

■ 2.4. Annäherung durch Dezimalbrüche
€ Approche par fractions décimales

Falls irgendwo in einem Ausdruck ein Dezimalpunkt vorkommt, gilt der ganze Ausdruck als Fliesskommazahl.
• S'il y a quelque part dans une expression un point décimal, tout l'expression a la valeur d'un nombre à virgule flottante.

Probiere aus: • Essaie:

```
1/2 + 2.4/144
```

```
.5/7 + Pi
```

```
.5/7 + Pi // N
```

■ 2.5. Komplexe Zahlen
€ Nombres complexes

"I" bedeutet die imaginäre Einheit. "Re" erzeugt den Realanteil, "Im" den Imaginäranteil, "Abs" den absoluten Betrag, "Conjugate" das konjugiert Komplexe, "Round" rundet. Schaue, was *Mathematica* tut und überlege wieso das Resultat so aussieht:
• "I" signifie l'unité imaginaire. "Re" crée la partie reelle, "Im" la partie imaginaire, "Abs" la valeur absolue, "Conjugate" le complexe conjugé, "Round" arrondit. Observe ce que *Mathematica* fait et réfléchis pourquoi le résultat est ainsi:

```
(2 + 4I)/(5 + 2I)

Re[3 + 4I]

Im[3 + 4I]

Conjugate[3 + 4I]

Abs[3 + 4I]

Round[2.7 - 8.6 I]

Cos[3 + 4I] // N

Sin[3 + 4I]

Exp[3 + 4I] // N

Log[3 + 4I]
```

■ 2.6. Symbole und Zahlen, Listen etc.
€ Symboles et nombres, listes etc.

Für *Mathematica* ist ein Buchstabe oder eine Buchstabenfolge ein Symbol, z.B. eine Variable. Variablen können auch mit "$" beginnen. "7a" jedoch bedeutet "7 mal a". Probiere aus:
• Pour *Mathematica* une lettre ou une suite de lettres sont un symbole, p.ex. une variable. Les variables peuvent aussi commencer par "$". "7a" par contre signifie "7 fois a". Essaie:

```
7a

a = 5

7a
```

In "Miscellaneus/Units.nb" sind Einheiten und Funktionen zur Einheitenverwandlung gespeichert:
• Dans "Miscellaneos/Units.nb" sont stockées les unités et les fonctions pour transformer les unités:

```
Needs["Miscellaneous`Units`"]

120 Pound

Convert[120 Pound, Kilogram]
```


Wichtige transzendente Zahlen sind als Symbole gespeichert.
Probiere aus:
• Les nombres transcendants importants sont stockés ou bien sauvés comme symboles.
Essaie:

```
E

N[E]

I

I^2

Degree

N[Sin[Pi/4]]

N[Sin[45 Degree]]
```

Mathematica kann weiter mit Listen (Vektoren, Matrizen, Tabellen etc rechnen). Der Themenkreis wird spÙter behandelt.
• *Mathematica* sait aussi calculer avec des listes (vecteurs, matrices tableaux etc.) Cet ensemble de thémes sera traité plus tard.

```
??List
```

■ 2.7. Approximationen
€ Approximations

Rationale Zahlen sind exakt gespeichert. "N" bedeutet approximieren. Probiere aus:
• Les nombres rationaux sont sauvés précisément. "N" signifie approximer. Essaie:

```
Sqrt[17]

N[%]

17^(1/2)

% // N

17^(0.5)

N[Sqrt[17]]

Sqrt[17] // N

Precision[%]
```

Wie gross ist also die Präzision? Die Präzision kann "beliebig" vergrðssert werden:
• Combien la précision est-elle grande? La précision peut être agrandie "quant on veut":

```
N[Sqrt[17], 100]
```

Wie gross ist hier die Präzision? • Combien la précision est-elle grande?

```
Pi

N[Pi]

N[Pi,500]

N[E,100]
```

■ 2.7. 1. Exakt versus approximiert:
€ Exact versus approximé:

Probiere aus: • Essaie:

```
3/10 + 1/2

Precision[%]

.36 + 1/3

Precision[%]
```

■ 2.7.2. Konvertierung approximativer Werte in exakte Werte:
€ Transformation des valeurs approximatives en valeurs exactes:

Lasse Dir mit der Help-Funktion folgende Funktionen erklären:
• Fais-toi expliquer par la fonction Help les fonctions suivantes:

```
?Rationalize

?Round

?Chop

?Floor

?Ceiling
```

Probiere aus: • Essaie:

```
Rationalize[3.1416]

Rationalize[3.1415926536]

Rationalize[3.1415926536,0]

Round[2.57432]

Round[2,45892]
```

Was war los ? • Que se passe-t-il?

```
Round[Sqrt[17]]

Round[N[Sqrt[17]]]

Chop[0.00000000002]
```



```
Ceiling[3.4]

Floor[3.4]
```

■ 2.7.3. Arbeiten auf einem selber fixierten Präzisions-Niveau:
€ Travailler à un niveau de précision fixé par soi-même:

Man kann z.B. selbst eine Funktion definieren, die die auszugebenden Kommastellen festlegt:
• On peut définir soi-même par exemple une fonction qui détermine le nombre de places dernière la virgule à emettre:

```
n30[x_]:=N[x,30]
```

Anwendung: • Application:

```
n30[5/7]

N[5/7]
```

Es gibt aber auch die globale Variable "$Post", in die sich eine Funktion eigeben lässt. Diese Variable wird auf jeden auszugebenen Ausdruck angewandt:
• Mais il existe aussi la variable globale "$Post", dans laquelle on peut entrer une fonction. Cette variable est appliqué à chaque expression à emettre:

```
?$Post

$Post=n30

Sqrt[3]

Precision[%]
```

Der Inhalt von $Post soll auch wieder gelöscht werden können. Das geschieht durch Eingabe des Zeichens för "missing value", nämlich "." :
• Il faut pouvoir aussi effacer le contenu de $Post. On peut faire cela en entrant le signe pour "missing value", c'est-à-dite "." :

```
$Post=.

Sqrt[3]

N[Sqrt[3]]

N[Pi,25]

N[%,200]
```

Wie genau ist also die übliche Präzision?
• Combien le précision courante est-elle exacte?

■ 2.8. Formatierte Zahlenausgabe
€ Sortie formatée de nombres

Mathematica bietet Mäglichkeiten, die Zahlenausgabe zu formatieren. So lässt sich die Leserlichkeit verbessern. Probiere aus:
Mathematica offre la possibilité de formater la sortie de nombres. $Ainsi on peut améliorer la lisibilité. Essaie:

```
Options[NumberForm]

1.23456789 10^20

NumberForm[1.23456789 10^20,
       ExponentStep -> 19,
       NumberSigns  -> {"-","+"}
           ]

NumberForm[1.23456789 10^20,
       ExponentStep -> 21,
       NumberSigns  -> {"-","+"}
           ]

NumberForm[123456789 10^20,
       ExponentStep -> 1000,
       NumberSigns  -> {"-","+"}
           ]
```

■ 2.9. Wichtige gespeicherte Konstanten
€ Constantes importantes stockées (sauvées)

Falls Dir eine der folgenden Konstanten unbekannt ist, so schlage in deinem Formelbuch nach. Probiere aus:
• Si tu ne connais pas l'une des constantes suivantes, consulte ton manuel de formules. Essaie:

```
??Catalan

?Degree

?E

?EulerGamma

?GoldenRatio

?I

?Pi

Pi

NumberForm[N[Pi,45],
       NumberSeparator  -> " ",
       DigitBlock -> 5
           ]

Log[E]
```



```
Cos[Pi/3]

N[Sin[45 Degree]]

avogadro = 6.02250 10^23

Log[avogadro]
```

■ 2.10. Zufallszahlen
€ Nombres aléatoires

Mathematica besitzt die Funktion "Random", die gleichverteilte Pseudozufallszahlen generiert. Solche braucht man z.B. in der Statistik, um Testdaten zu erzeugen. Probiere aus:
• *Mathematica* possède la fonction "Random", qui génère des nombres pseudo-aléatoires en distribution homogène. On les emploie p.ex. dans la statistique, pour créer les données de test. Essaie:

```
Random[]

Random[Integer]

Random[Integer, {0, 100}]

Random[Complex, {1 + 3I, 4 + 7I}]
```

Die Packages "ContinousDistributions.nb" und "DiscreteDistributions.nb" enthalten Definitionen, mit denen sich andere als gleichverteilte Pseudozufallszahlen erzeugen lassen. Probiere aus:
• Les Packages "ContinousDistributions.nb" et "DiscreteDistributions.nb" contiennent des définitions par lesquelles on peut générer des nombres pseudo-hasard autres que distribués homogènement. Essaie:

```
Needs["Statistics`ContinuousDistributions`"]

Random[NormalDistribution[5]]
```

```
3.402639897124415
```

Der Pseudozufallszahlengenerator benutzt die Zeit seit dem Einschalten der Maschine als Variable zur Berechnung der Zahlen. Mit "SeedRansom" lässt sich der Generator wieder "zurücksetzen". Das kann benutzt werden, um mehrmals dieselben Zahlen zu generieren. Probiere aus:
• Le générateur des nombres pseudo-aléatoire utilise le laps de temps dès le démarrage de la machine comme variable pour calculer les nombres. Par "SeedRansom" on peut "remettre en arrière" le générateur. On employe cela pour générer plusieurs fois les mêmes nombres. Essaie:

```
??SeedRandom

SeedRandom[2]

{Random[], Random[]}

SeedRandom[2]

{Random[], Random[]}
```

■ 2.11. Iteratoren

Mathematica stellt iterative Funktionen (Iteratoren) zur Verfügung, um die Eingabe
wiederholt er Rechnungen abzukürzen. Einige sind hier gezeigt. Probiere aus:
• *Mathematica* met à disposition des fonctions itératives pour abréger l'entrée de calculs répétés. Voici quelques-uns.
Essaie:

```
??Table

Table[x, {5}]

??Product

Product[x + k y, {k, 4}]

??Do

Do[Print["k: ", k], {k, 7, 9}]

??TableForm

TableForm[Table[{k, 10^(-k), Chop[N[10^(-k)]]}, {k, 7, 14}]]

??Sum

Sum[i x^i, {i, 3, 23, 5}]
```

■ 2.12. Matrizenrechnung
€ Calcul des matrices

Probiere aus: • Essaie:

```
??MatrixForm

??Inverse

??Det

??Eigenvalues

??Eigenvectors

??LinearSolve

??NullSpace

??RowReduce

??Transpose

??IdentityMatrix

m1 = {{1, 2},{3, 4}}

Transpose[m1] // MatrixForm
```



```
MatrixForm[m1]

MatrixForm[{{1, 2},{3, 4}}]

m2 = {{0, -1},{8, -6}}

m3 = {{0, -1},{8, -6}} // MatrixForm
```

Das Matrixprodukt wird mit einem gewöhnlichen Punkt geschrieben. Probiere aus:
• Le produit de matrices s'écrit avec un simple point. Essaie:

```
m1.m2 // MatrixForm

m1.m3
```

Elementweise Multiplikation. Probiere aus:
• Multiplication par élément. Essaie:

```
m1 m2 // MatrixForm
```

Probiere aus: • Essaie:

```
?NullSpace

m4 = {{1,2,3},{1,2,3},{0,0,1}}

NullSpace[m4]

?RowReduce

RowReduce[m4] // MatrixForm

Eigenvectors[m1]

Transpose[m1] // MatrixForm

m = Table[Random[], {3}, {3}]

MatrixForm[m]

mInv = Inverse[m]

MatrixForm[%]

m.mInv

MatrixForm[%]

Chop[%]

MatrixForm[%]

IdentityMatrix[7] // MatrixForm
```

■ 2.13. Lösen von Gleichungen
€ Résoudre des équations

Probiere aus: • Essaie:

```
??Solve
```

```
??NRoots
```

```
??FindRoot
```

```
Solve[{2x - 3y ==8, 4x + y == 6},{x, y}]
```

```
Solve[m1.{{x},{y}}=={{2},{5}}, {x, y,}]
```

```
NRoots[x^4 +3 x^2 + 5x == 7, x]
```

FindRoot benutzt die Newton-Methode. Darin wird die Ableitung benutzt:
• FindRoot emploie la méthode de Newton. On y employe la déduction:

```
FindRoot[Sin[x]/x == 0, {x, 2}]
```

Bessel-Funktionen: • Fonctions de Bessel

```
Plot[BesselJ[0, x], {x, 0, 12}];
```

```
FindRoot[BesselJ[0, x]/x == 0, {x, 12}]
```

```
FindRoot[BesselJ[0, x]/x, {x, 12}]
```

```
FindRoot[BesselJ[0, x]/x, {x, 7}]
```

```
Plot[BesselJ[0, x], {x, 0, 15}];
```

FindRoot benutzt die Newton-Methode. Darin wird die Ableitung benutzt. Hier ein Plot der Ableitung obiger Bessel-Funktion. (Beachte, dass die Ableitung bei x = 7 klein ist.)
• FindRoot emploie la méthode de Newton. On y employe la déduction. Voici un plot de la déduction de la fonction de Bessel ci-dessus. (Remarque que la déduction est petite pour x = 7.)

```
Plot[-BesselJ[1, x], {x, 0, 15}];
```

■ 2.14. Numerische Integration etc.
€ Intégtation numérique etc.

Probiere aus: • Essaie:

```
??NIntegrate
```

```
NIntegrate[Sin[x^2]/x^2, {x, 0, 1}]
```

```
Plot[Sin[x^2]/x^2, {x, 0, 1}];
```

```
NIntegrate[Sin[Sin[Cos[x^10]]], {x, 0, 1}]
```



```
??NSum
```

```
NSum[(n^2 - n + 1) Log[n]/n^3, {n, 1, 100}]
```

```
??NProduct
```

```
NProduct[(1 + 1/n)^n, {n, 1, 100}]
```

```
Limit[(1 + 1/n)^n, n->Infinity]
```

■ 2.15. Numerische Lösung von Differentialgleichungen
€ Solution numérique d'équations différentielles

Probiere aus: • Essaie:

```
??NDSolve
```

```
NDSolve[{y''[x] - y'[x] + x y[x] == 0, y'[0] == -1, y[0] == 1},
       y, {x, 0, 5}]
```

```
Plot[y[x] /. %, {x, 0, 5}];
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

2. Numerische Probleme
€ Problèmes numériques

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 2 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 2.
WIR94/99

■ Aufgabe 1 € Problème 1

Berechne die Anzahl Minuten in einem Jahr mit 365 Tagen.
• Calcule le nombre de minutes pour une année de 365 jours.

```
365 Tag (24 Std/Tag) (60 Min/Std)
```

■ Aufgabe 2 € Problème 2

■ (a)

Berechne Pi auf 770 Stellen. Findest Du 6 sich folgende Ziffern 9?
• Calcule Pi sur 770 places. Trouves-tu 6 fois de suite le chiffre 9?

```
N[Pi,770]
```

■ (b)

Wie nahe ist e hoch (Pi mal Wurzel aus 163) bei einer ganzen Zahl?
• Combien de près d'un nombre sntiers se trouve e puissance (Pi fois racine de 163)?

```
N[E^(Pi 163^(1/2)),100]-Ceiling[N[E^(Pi 163^(1/2)),100]]
```

```
N[E^(Pi 163^(1/2)),100]-Floor[N[E^(Pi 163^(1/2)),100]]
```


■ Aufgabe 3 € Problème 3

Verwende NumberForm um die Zahl 123456789 in 3-er-Blöcken auszugeben.
• Sers-toi de NumberForm pour sortir le nombre 123456789 en blocs de 3 chiffres.

```
NumberForm[123456789, DigitBlock->3]
```

■ Aufgabe 4 € Problème 4

Generiere mit Hilfe von Random zwei Pseudozufallszahlen und addiere diese.
• Générer à l'aide de Random deux nombres pseudo-aléatoire et fais-en l'addition

```
a=Random[];b=Random[];a+b
```

■ Aufgabe 5 € Problème 5

Schaue, was *Mathematica* tut, und überlege wieso das Resultat so aussieht:
• Observe ce que fait *Mathematica* et réfléchis pourquoi le résultat est ainsi:

```
Sqrt[3] + 3
```

```
Exp[2 Pi I]
```

```
5 > 3
```

```
Pi^E > E ^ Pi
```

```
N[Pi^E > E ^ Pi]
```

■ Aufgabe 6 € Problème 6

■ (a)

Studiere die folgenden Befehle:
• Etudie les ordres suivants:

```
??Table
```

```
??Sum
```

```
??Product
```

■ (b)

Kreiere eine Liste mit 50 Ziffern 9.
• Crée une liste avec 50 fois le chiffre 9.

```
Table[9,{50}]
```

■ (c)

Kreiere eine Liste mit den ersten 24 Quadratzahlen.
• Crée une liste avec 24 premiers nombres carrés.

```
Table[i^2,{i,24}]
```

■ (d)

Zeige mit Sum, dass 100 * 101/2 die Summe der ersten 100 natürlichen Zahlen ist.
• Montre par Sum que 100 * 101/2 est la somme de 100 premiers nombres naturels.

```
s = Sum[i,{i,100}]

p = 100 101 / 2

s == p
```

■ (e)

Berechne 1+1/1!+.....+1/10!
• Calcule 1+1/1!+.....+1/10!

```
Sum[1/(n!),{n,0,10}]
```

■ (f)

Berechne 6! mit "Product".
• Calcule 6! avec "Product".

```
Product[n,{n,1,6}]

6!
```

■ Aufgabe 7 € Problème 7

Berechne die Inverse der 3*3-Hilbert-Matrix. ((i,j) = 1/(i+j-1).)
• Calcule l'inverse de la matrice de Hilbert 3*3. ((i,j) = 1/(i+j-1).)

```
h = Table[1/(i+j-1),{i,3},{j,3}]; MatrixForm[h]

MatrixForm[Inverse[h]]
```


■ Aufgabe 8 € Problème 8

Berechne eine numerische Approximation von Pi durch Integration der Funktion
4/(1+x^2) im Intervall [0,1]
• Calcule une approximation numérique de Pi en intégrant la fonction 4/(1+x^2) dans l'intervalle [0,1]

```
f[x_]:=4/(1+x^2); f[x]

Integrate[f[x],x]

Integrate[f[x],{x,0,1}]

NIntegrate[f[x],{x,0,1}]
```

■ Aufgabe 9 € Problème 9

■ Finde 5 Wurzeln der folgenden Gleichungen € Trouve 5 racines de l'équation suivante

■ (a) (NRoots geht für Polynome) € (NRoots joue pour des polynômes)

x^5+5x^4+4x^3+3x^2+2x+1=0

```
NRoots[x^5+5x^4+4x^3+3x^2+2x+1==0,x]
```

■ (b) (FindRoot geht für allgemeine Funktionen, liefert jedoch jeweils nur eine Nullstelle) € (FindRoot va pour des fonctions générales, mais ne livre qu'un zéro)

sin^2(Pi x) - x^2 cos(Pi x) = x

```
Table[FindRoot[Sin[Pi x]^2 - x^2 Cos[Pi x] == x,
{x,x0}],{x0,-5,5}]
```

■ (c) (Wie in b) € (Comme dans b)

x tan(x) = 1

```
Table[FindRoot[x Tan[x] == 1,
{x,x0}],{x0,13}]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

3. Algebraische und symbolische Stärken
€ Forces algébriques et symboliques

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 3 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 3.
WIR94/99

■ 3.1. Algebra
€ Algèbre

Mathematica kann algebraische Ausdrücke manipulieren.
• *Mathematica* peut manipuler des expressions algébriques

■ 3.1.1. Studiere: € Etudie:

```
?Apart
?Cancel
?Collect
?Expand
?ExpandAll
?ExpandDenominator
?ExpandNumerator
?Factor
?Simplify
```



```
?Short
?Together
?TrigExpand
Needs["Algebra`Trigonometry`"]
?TrigExpand
?TrigFactor
?TrigReduce
?ComplexToTrig
?TrigToComplex
?Sum
Sum[i^2,{i, n}]
Needs["Algebra`GosperSum`"]
?GosperSum
```

Was in alten Versionen einmal existiert hat, braucht in neuen nicht mehr zu existieren...
• Ce qui esistait dans des vieilles versions, n'existe pas forcement dans des versions nouvelles.

```
$Version (*Mathematica-Version*)
?$Version
Needs["Algebra`SymbolicSum`"]
Sum[i^2,{i, n}]
?Limit
?Integrate
?Series
?DSolve
??DSolve
```

■ 3.1.2. Probiere: € Essaie:

Ueberlege Dir, was durch die folgenden Operationen bewirkt wird!
• Réfléchis à ce qui est produit par les opérations suivantes!

```
Expand[(x + 2y + z)^4]
Factor[%]
Factor[6x^3 + 35 x^2 y + 58 x y^2 + 21 y^3]
```

■ 3.2. Gleichungen lösen
€ Résoudre des équations

■ 3.2.1. Algebraisch exakte Lösung:
€ Solution exacte algébriquement:

NRoots rechnet eine Näherung. "||" bedeutet dabei das logische "oder", "or".
• NRoots calcule une approximation. "||" signifie le "ou" logique.

```
NRoots[x^3 - 3 x^2 - 17 x + 51 == 0, x]
```

"||" ist das logische "oder".
• "||" est le "ou" logique.

```
??||
```

Gewisse algebraische Gleichungen oder Gleichungssysteme kann man jedoch exakt lösen. Dies geschieht mit "Solve".
• On peut résoudre exactement certains équations algébriques ou certains systèmes d'équations. Cela avec "Solve".

```
Solve[x^3 - 3 x^2 - 17 x + 51 == 0, x]
```

Angenähert: • Approximé

```
% // N
```

Uebersichtlicher dargestellt:
• Représenté d'une façon plus claire:

```
ColumnForm[Solve[{  x y + 5x + 6y == 7,
                    x^2 + 5x + 7y == 8}]]
```

Der Output besteht aus einer Menge von Ersetzungsregeln.
Wie lässt sich damit weiterrechnen?
• L'output est composé d'une quantité de règles de remplacement. Comment continue-t-on de calculer avec cela?

■ 3.2.2. Weiterverwendung der Lösung:
€ Réutilisation de la solution:

"/." bedeutet "ersetze jedes" resp. "ReplaceAll".
• "/." signifie "remplace tout" resp. "Remplace All".

```
?/.
```

```
?ReplaceAll
```

```
?Replace
```

Studiere die folgenden Anwendungen:
• Etudie les applications suivantes:


```
a = x /. x -> 7
```

```
a
```

```
x
```

Weiter: • Ensuite:

```
x + 5y /. {x -> 1, y -> 2/7}
```

```
x + 5y
```

Weiter: • Encore:

```
Solve[2 x^5 + 12 x^2 + 18x - 14 == 0]
```

Die Lösung hier ist nur symbolisch. Eine allgemeine algebraisch exakte Lösungsformel
für Gleichungen mit Polynomen 5. Grades existiert nicht. Approximativ allerdings geht es:
• La solution n'est que symbolique. Une forme de solution algébrique générale exacte pour les équations avec polynômes du 5e degré n'existe pas. Mais une approximation existe:

```
N[%]
```

■ 3.3. Vereinfachungen
€ Simplifications

■ 3.3.1. Allgemeines
€ Généralités

Mathematica kann nicht die Frage beantworten, welche Outputform eines Ausdrucks für den Verwender der einfachste ist. Man muss den Ausdruck selbst in die gewünschte Form bringen. Dafür gibt es Hilfsmittel wie z.B. "Simplify":
• *Mathematica* ne sait pas répondre à la question, laquelle des formes de l'Output d'une expression est la plus simple pour l'utilisateur. Il faut donner soi-même la forme souhaitée à l'expression. Pour cela il y a des moyens tels que p.ex. "Simplyfy".

```
Solve[(a d x^3 + 14 b d x^2 + 14^2 c d x)
                == 0, x] (14b + 14a)/(-c + b)
```

```
Simplify[%]
```

```
??Simplify
```

```
x/((x+2)(x-2))
```

```
Simplify[%]
```

```
x/Expand[((x+2)(x-2))]
```

```
Expand[x/((x+2)(x-2))]
```

```
x/Simplify[((x+2)(x-2))]
```

■ 3.3.2. Andere Vorgehensweisen:
€ Autres façons de procéder

Beispiel: Partialbruchzerlegung mit "Apart". Probiere aus.:
• Exemple: Décomposition de fractions partielles en utilisant "Apart":

```
x/((x+2)(x-2))
```

```
Apart[%]
```

Retour: • Retour:

```
Together[%]
```

Man beachte die Reihenfolge der Terme im Output. Zuerst Konstanten, dann Symbole, alphabetisch, Terme nach steigendem Grad etc.:
• Il faut tenir compte des termes dans le output. D'abort les constantes, puis les symboles, en ordre alphabétique, termes selon le dégré en augmentant etc.:

```
Expand[(1 + 2y + 3z)^3]
```

Umordnen: • Réordonner:

```
Collect[%, y]
```

■ 3.3.3. Trigonometrische Ausdrücke:
€ Expressions trigonométriques:

```
Expand[Sin[x]/Cos[x]]
```

```
Expand[(Sin[x]+Cos[x])/Sin[x] (Sin[x]^2 + Cos[x]^2), Trig -> False]
```

```
?Trig
```

```
Expand[(Sin[x]+Cos[x])/Sin[x] (Sin[x]^2 + Cos[x]^2), Trig -> True]
```

```
Needs["Algebra`Trigonometry`"]
```

```
TrigToComplex[Sin[x]]
```

■ 3.3.4. Lange Resultate:
€ Résultats longs

Mathematica bietet die Möglichkeit, Resultate abzukürzen:
• *Mathematica* offre la possibilité d'abréger les résultats:

```
Short[Expand[(x + 2y + 3z)^9]]
```

```
?Short
```


Mit der globalen Variablen "$PrePrint" lässt sich das Abkürzen der Ausgabe generalisieren:
• Avec la variable globale "$PrePrint" on peut généralement abréger la sortie:

```
$PrePrint = Short
```

```
Expand[(x + 2y + 3z)^9]
```

```
Expand[(x + 2y + 3z)^11]
```

```
Expand[(x + 2y + 3z)^5]
```

Trotzdem kann man das ganze Resultat abrufen mit "Print":
• En tout cas on peut appeler le résultat entier avec "Print":

```
Print[%]
```

Alter Zustand: • Ancien état:

```
$PrePrint = .
```

```
Expand[(x + 2y + 3z)^9]
```

■ 3.4. Summation
€ Sommation

Mathematica bietet die Möglichkeit, z.B. Summen mit variablem oberen Summationsindex zu berechnen:
• *Mathematica* offre la possibilité p.ex. de calculer des sommes avec un index de sommation supérieur variable:

```
Needs["Algebra`SymbolicSum`"]
```

```
Sum[i^3,{i, n}]
```

Probiere eigene Aufgaben aus:
• Essaie des problèmes que tu inventes toi-même:

■ 3.5. Calculus (Differential- und Integralrechnung)
€ Calculus (Calcul différentiel et intégral)

■ 3.5.1. Unbestimmte Integrale
€ Intégrals intéterminées

Beispiele: • Exemple:

```
Integrate[Cos[x], x]
```

```
Integrate[x^4 Cos[x], x]
```

```
1/(2 + 3 x^2)^3
```

```
Integrate[1/(2 + 3 x^2)^3, x]
```

Kontrolle: Differenzieren!
• Controlle: Différencier!

```
D[%, x]
```

Das Resultat kann durchaus anders aussehen. Es muss vielleicht umgeformt werden.
• Le résultat peut être différent. Il faut peut-être le transformer.

```
Simplify[%]
```

■ 3.5.2. Bestimmte Integrale
€ Intégrales déterminées

Mathematica kann bestimmte Integrale exakt rechnen. Probiere aus:
• *Mathematica* sait calculer exactement des intégrales déterminées. Essaie:

```
?Integrate
```

```
?*Integ*
```

```
Integrate[Exp[x],{x, -1, 1}]
```

```
N[%]
```

Numerisch geht es natürlich auch. Probiere aus:
• Il va de soi qu'on peut le faire numériquement. Essaie:

```
NIntegrate[Exp[x],{x, -1, 1}]
```

Zu gewissen Funktionen kann keine einfache Stammfunktion gefunden werden:
• Pour certains fonctions on ne trouve pas de fonction primitive (intégrale) simple:

```
Integrate[Sin[Sin[x]],{x, 0, 1}]
```

Numerisch geht es natürlich. Probiere aus:
• Numériquement ça va. Essaie:

```
NIntegrate[Sin[Sin[x]],{x, 0, 1}]
```

■ 3.5.3. Integrale von Funktionen mit Polen
€ Intégrales de fonctions avec pôles

Bei Funktionen mit Polen können Fehler auftreten, wenn man einfach eine Stammfunktion bestimmt und die Grenzen einsetzt. Prüfe:
• Lors de fonctions avec pôles, des erreurs peuvent appataître, si on détermine simplement une fonction primitive (intégrale) et fixe les limites. Examine:

```
Integrate[1/(x-Pi/100)^2,{x, -1, 1}]
```



```
Plot[1/(x-Pi/100)^2,{x, -1, 1}];
```

```
NIntegrate[1/(x-Pi/100)^2,{x, -1, 1}]
```

```
Integrate[1/x^2,{x, -1, 1}]
```

```
NIntegrate[1/x^2,{x, -1, 1}]
```

```
??NIntegrate
```

Es kann z.B. angegeben werden, dass bei 0 eine Singularität (Polstelle) ist. Prüfe:
• On peut indiquer p.ex. qu'il y a une singularité (pôle) prèes de 0. Examine:

```
NIntegrate[1/x^2,{x, -1, 0, 1}]
```

■ 3.5.4. Mehrfachintegrale
€ Intégrales multiples

f(x, y) kann über x und y integriert werden. Probiere:
• f(x, y) peut être intégré sur x et y. Essaie:

```
Integrate[x^2 Sin[y],x, y]
```

Bei bestimmten Integralen sind die äusseren Grenzen zuerst anzugeben:
• Pour certains intégrales les limites extérieurs doivent êtres données d'abort.

```
Integrate[x^2 Sin[y],{x, 0, Pi/3}, {y, 0, x}]
```

■ 6. Grenzwerte
€ Valeurs limites

Mit "Limits" lassen sich Grenzwerte berechnen. "Unendlich" ist "Infinity":
• Avec "Limits" on peut calculer des valeurs limites. "Infinie" est "Infinity":

```
?Limit
```

```
?Infinity
```

```
Limit[Sin[x]/x, x -> 0]
```

```
Limit[1/x, x -> 0]
```

Der Wert des Grenzwerts kann davon abhängen, aus welcher Richtung man sich dem problematischen Punkt nähert. Die Richtung kann angegeben werden:
• La valeur de la valeur limite peut dépendre de la direction de laquelle on s'approche du point problématique.

```
Limit[1/x, x -> 0, Direction -> 1]
```

```
Limit[1/x, x -> 0, Direction -> -1]
```

```
Integrate[E^(-s t) t^n,{t,0,Infinity}]
```

```
Integrate[E^(-s t) Sin[w t],{t,0,Infinity}]
```

■ 7. Potenzreihen
€ Séries de puissances

Beispiele von Potenzreihenentwicklungen:
• Exemples de développements de séries de puissances:

```
Series[f[x],{x,0,9}]

Series[f[x],{x,1,6}]

Series[E^x,{x,0,9}]

Series[Cos[x],{x,0,9}]
```

Solch ein Output ist noch kein algebraischer Ausdruck, mit dem weitergerechnet werden
kann, denn die angegebene Ordnung der nachfolgenden Glieder ist kein Term. Mit "Normal" kann diese Angabe entfernt werden. Probiere:
Un tel Output n'est pas encore une expression algébrique, avec laquelle on peut continuer de calculer, car l'ordre donné des membres suivantes n'est pas un terme. On peut effacer ce problème par "Normal". Essaie:

```
a = Normal[Series[E^x,{x,0,9}]]

b = Normal[Series[Cos[x],{x,0,9}]]

b = Normal[Series[Cos[x],{x,0,9}]]

b /. x->1

a + b
```

Etwas fürs Auge: • Quelque chose pour l'oeil:

```
Clear[a]

prCos[x_]:= Normal[Series[Cos[x],{x,0,11}]];
p = prCos[x]

Plot[{p, Cos[x]}, {x, 0, 2 Pi}];

a = Append[Evaluate[Table[Normal[Series[Cos[x],{x,0,n}]],{n, 12}]],
Cos[x]]

Plot[Evaluate[Table[a[[i]],{i,13}]], {x, 0, 2 Pi}];

??Append
```

■ 8. Lösen von Differentialgleichungen
€ Résoudre des équations différentielles

Mathematica kann gewisse einfache Differentialgleichungen exakt lösen. Probiere:
• *Mathematica* peut résoudre avec exactitude certains équations différentielles. Essaie:

```
?DSolve
```



```
DSolve[y''[x] + y'[x] + y[x] == 0, y[x], x]
```

"C[..]" bedeutet eine Konstante. Auch Systeme können behandelt werden. Probiere:
• "C[..]" signifie une constante. On peut aussi traiter des systèmes. Essaie:

```
d = DSolve[{y'[x] + y[x] == 0, y[2] == 2}, y[x], x]

dd =Flatten[d][[1]]

ddd = dd[[2]]

Plot[ddd, {x, -3, 3}];
```

Aufgabe:Behandle das System {r y'[x] + y[x] == 0, y[2] == 2} für
r = -3, -2, -1, 0, 1, 2, 3 und vergleiche die Graphen.
• Problème: Traite le système {r y'[x] + y[x] == 0, y[2] == 2} pour
r = -3, -2, -1, 0, 1, 2, 3 et compare les graphes.

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

3. Algebraische und symbolische Stärken
€ Forces algébriques et symboliques

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 3 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 3.
WIR94/99

■ Aufgabe 1 € Problème 1

■ (a)

Löse folgende Gleichung: • Résous l'équation suivante:

```
Solve[x^2 + 2x + 1 == 0]
```

■ (b)

Löse folgende Gleichung: • Résous l'équation suivante:

```
Solve[ax^2 + bx + c == 0,x]
```

■ (c)

Löse folgende Gleichung: • Résous l'équation suivante:

```
Solve[{x + y== 5, 2x + 6y == 23}]
```


■ Aufgabe 2 € Problème 2

■ (a)

Faktorisiere das Polynom 1-x^n. Entdeckst Du ein Gesetz?
• Factorise le pôlynome 1-x^n. Découvres-tu une loi?

```
Do[Print[1 - x^n,"==",Factor[1 - x^n]],{n, 2, 9}]
```

■ (b)

Faktorisiere das Polynom 1-x^11
• Factorise le pôlynome 1-x^11.

```
Factor[1 - x^11]
```

■ (c)

Faktorisiere das Polynom 1-x^..... (Zahl einsetzen!)
• Factorise le pôlynome 1-x^..... (Mettez un nombre!)

```
Factor[1 - x^13]
```

■ Aufgabe 3 € Problème 3

■ (a)

Mache mit "Apart" eine Partialbruchzerlegung:
• Fais avec "Apart" une décomposition d'une fraction rationelle en éléments simples:

```
Apart[x/(x^2 + 5x + 6)]
```

■ (a)

Mache mit "Apart" eine Partialbruchzerlegung:
• Fais avec "Apart" une décomposition d'une fraction rationelle en éléments simples:

```
Apart[(2x + 7)/(x^3 + 3x^2 + 3x + 1)]
```

■ Aufgabe 4 € Problème 4

■ (a)

Vereinfachung trigonometrischer Funktionen: Setzt Trig->True:
• Simplifications de fonctions trigonométriques: Mettez Trig->True:

```
Expand[Cos[x] Cos[y] + Sin[x] Sin[y],Trig->True]
```

```
Factor[Cos[x] Cos[y] + Sin[x] Sin[y],Trig->True]
```

■ (b)

Vereinfachung trigonometrischer Funktionen: Setzt Trig->True:
• Simplifications de fonctions trigonométriques: Mettez Trig->True:

```
Expand[Cos[x] Sin[y] + Sin[x] Cos[y],Trig->True]
```

```
Expand[Cos[x] Sin[y] + Sin[x] Cos[y],Trig->True]
```

■ (c)

Vereinfachung von Exponentialfunktionen: Setzt Trig->True:
• Simplifications de fonctions esponentielles: Mettez Trig->True:

```
Expand[E^(-I x) + E^(I x),Trig->True]
```

```
Factor[E^(-I x) + E^(I x),Trig->True]
```

■ Aufgabe 5 € Problème 5

■ (a)

Generiere mit "Array" eine 2 x 2-Matrix mit den Elementen b[i, j]:
• Génère avec "Array" une matrice 2 x 2 avec les éléments b[i, j]:

```
Clear[b]; m = Array[b,{2,2}]
```

■ (b)

Invertiere und Transponiere m:
• Invertis et transforme m:

```
Inverse[m]
```



```
Transpose[m]
```

■ (c)

Generiere mit "Array" eine 2 x 2-Matrix mit den Elementen c[i, j]:
• Génère avec "Array" une matrice 2 x 2 avec les éléments c[i, j]:

```
Clear[c]; n = Array[c,{2,2}]
```

```
MatrixForm[n]
```

■ (d)

Berechne m n und m.n . Was ist der Unterschied?
• Calcule m n et m.n . Quelle est la différence?

```
m n
```

```
m.n
```

```
MatrixForm[m n]
```

```
MatrixForm[m.n]
```

■ Aufgabe 6 € Problème 6

■ (a)

Integration eines Ausdrucks:
• Intégration d'une expression:

```
w = 1/(x^3 + 1)
```

```
u = Integrate[1/(x^3 + 1),x]
```

■ (b)

Differenziere das Resultat:
• Différencie le résultat:

```
v = D[%,x]
```

■ (c)

Hat man nun wieder den ursprünglichen Ausdruck?
• A-t-on de nouveau l'expression d'origine?

```
Together[v]

v=ExpandAll[Together[v]]

w === v
```

■ Aufgabe 7 € Problème 7

■ (a)

Exakter Wert eines Integrals :
• Valeur exacte d'un intégrale:

```
Integrate[x y^2,{x,0,1},{y,0,Sqrt[1-x]}]
```

■ (a)

Exakter Wert eines Integrals :
• Valeur exacte d'un intégrale:

```
Integrate[y Exp[x^2],{x,0,1},{y,0,Sqrt[x]}]
```

■ Aufgabe 8 € Problème 8

■ (a) Potenzreihenentwicklung € Développeer de séries de puissances

```
Series[E^x ,{x, 0, 12}]
```

■ (b) Potenzreihenentwicklung € Développeer de séries de puissances

```
Series[1/(1 - x),{x, 0, 8}]
```

■ (c) Potenzreihenentwicklung € Développeer de séries de puissances

```
Series[1/(1 - x)^2, {x, 0, 10}]
```

■ (d) Potenzreihenentwicklung € Développeer de séries de puissances

```
Series[f[x], {x, 0, 10}]
```


■ Aufgabe 9 € Problème 9

■ (a) Interessante Potenzreihenentwicklung € Développement intéressant de séries de puissances

```
Series[1/(1 +  x) ,{x, 0, 7}]
```

■ (b) Interessante Potenzreihenentwicklung € Développement intéressant de séries de puissances

```
Series[1/(1 - 2x),{x, 0, 8}]
```

■ (c) Interessante Potenzreihenentwicklung € Développement intéressant de séries de puissances

```
(1+ 2x + 3x^2) Series[1/(1 - x^3), {x, 0, 8}]
```

■ Aufgabe 10 € Problème 10

■ (a)

Probleme mit Integration:
• Problème d'intégration

```
Integrate[1/(1 + Sin[x]^2 Sqrt[Pi^2 + x]),{x,-2,2}]
```

■ (b)

Näherung durch Integration einer Potenzreihe:
• Approximation par l'intégration d'une série de puissances:

```
Clear[s];
s = Series[1/(1 + Sin[x]^2 Sqrt[Pi^2 + x]),{x,0,8}]
```

Verwandlung in einen verwertbaren Ausdruck (O[...] weglassen)
• Transformation en une expression utilisable (omettre O[...])

```
sn = Normal[s]
```

Jetzt kann man integrieren:
• Maintenant on peut intégrer:

```
Integrate[sn,{x,-2,2}]
```

Numerisch: • Numériquement:

```
N[%]
```

■ Aufgabe 11 € Problème 11

■ **(a) Differentialgleichung € Equation différentielle**
y'(x) - y(x) tan x = x

```
DSolve[y'[x] - y[x] Tan[x]== x, y[x], x]
```

■ **(b) Differentialgleichung € Equation différentielle**
y'(x) + y(x) tan x = sec(x) (secans)

```
DSolve[y'[x] + y[x] Tan[x]== Sec[x], y[x], x]
```

■ **(c) Differentialgleichung € Equation différentielle**
y'(x) = y(x)
mit Randwertbedingung
€ avec condition de valeur au bord
y'(0)=1

```
Clear[y];
DSolve[{y'[x] == y[x] , y'[0] == 1}, y[x], x]
```

■ **(d) Differentialgleichung € Equation différentielle**
y''''(x) - k^4 y(x) = 0
(Vibration eines Strahls € Vibration d'un rayon)

```
DSolve[y''''[x] - k^4 y[x] == 0, y[x], x]
```

■ **(b) Differentialgleichung € Equation différentielle**
y'(x) + y(x) tan x = sec(x) (secans)

```
DSolve[y'[x] y[x] + x^2 == x, y[x], x]
```

■ "Putzmaschine" einsetzen € Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Kurs € Cours

4. Graphiken € Graphiques

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 4 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 4.
WIR94/99

■ 4.1. Zweidimensionale Plots € Plots à deux dimensions

Gewöhnliche Funktion über einem Intervall. ";" unterdrückt den letzten Output:
• Fonction commune sur un intervalle. ";" empêche le dernier output.

```
Plot[x^2 + 5x + 6,{x, -10, 5}];

Plot[x^2 + 5x + 6,{x, -10, 5}]

y = x^2 + 5x + 6;
Plot[y,{x, -10, 5}];
```

Pole bieten keine Probleme:
• Les pôles ne causent pas de problèmes:

```
Plot[Tan[x],{x, 0, 10}];
```

Vielleicht wird etwas abgeschnitten:
• *Mathematica* coupe peut-être quelque chose:

```
Plot[Sin[x]/x,{x, -20, 20}];
```

■ 4.2. Options
€ Options

■ 4.2.1. Allgemeines
€ En général

Ein Plot kann auf diverse Weise durch Angabe der Options verändert werden:
• Un plot peut être transformé de différentes façons en indiquant les options:

```
??Plot
```

```
Options[Plot]
```

Beispiel mit dem y-Intervall:
• Exemple avec l'intervalle y:

```
Plot[Sin[x]/x,{x, -20, 20}, PlotRange -> {0, 1}];
```

```
Plot[Sin[x]/x,{x, -20, 20}, PlotRange -> All];
```

Vielleicht wird trotzdem "etwas" abgeschnitten...:
• Peut-être *Mathematica* coupe quand-même "quelque chose"...:

```
Plot[Cot[x],{x, 1, 10}, PlotRange -> All];
```

```
Plot[Cot[x],{x, 0, 10}, PlotRange -> All];
```

```
Plot[Cot[x],{x, -10, 10}, PlotRange -> All];
```

■ 4.2.2. Die Voreinstellung von Mathematica kann abgefragt werden:
€ La mise au point préalable de *Mathematica* peut être interrogée:

```
Options[Plot]
```

```
AspectRatio /. Options[Plot]
```

```
??AspectRatio
```

```
PlotPoints /. Options[Plot]
```

Der Output kann unterdrückt werden:
• On peut réprimer l'output.

```
test = Plot[Sin[x],{x, 0, 2 Pi},
DisplayFunction -> Identity];
```

Hier eine Zeichnung, die zeigt, in welchen Abständen *Mathematica* die Funktionswerte berechnet:
• Voici un dessin qui montre à quelles distances *Mathematica* calcule les valeurs des fonctions:


```
Show[Graphics[{Thickness[0.001], Map[Line[{{#[[1]],
    0},#}]&, Nest[First, test, 4]]}],
    Axes->Automatic];
```

Was tun die einzelnen Befehle?
• Que fait chaque ordre?

```
test
```

```
??Nest
```

```
Nest[First, test, 4]
```

```
Nest[First, test, 5]
```

```
Nest[First, test, 3]
```

```
Nest[First, test, 2]
```

```
Nest[First, test, 1]
```

```
First[test]
```

```
Map[Line[{{#[[1]],0},#}]&, Nest[First, test, 4]]
```

```
Graphics[{Thickness[0.001],
                Map[Line[{{#[[1]],0},#}]&,
                Nest[First, test, 4]]}]
```

```
Show[Graphics[{Thickness[0.001],
    Map[Line[{{#[[1]],0},#}]&,
    Nest[First, test, 4]]}], Axes->Automatic];
```

■ 4.2.3. Probleme mit den Plot-Punkten:
€ Problèmes avec les points-"plot":

Kann das stimmen?
• Est-ce correct?

```
Plot[x + Sin[2 Pi x], {x, 0, 24}];
```

Andere Anzahl Punkte:
• Autre nombre ce points:

```
Plot[x + Sin[2 Pi x], {x, 0, 24}, PlotPoints -> 20];
```

```
Plot[x + Sin[2 Pi x], {x, 0, 24}, PlotPoints -> 50];
```

Welche Punkte rechnet *Mathematica*?
• Quels points sont calculés pat *Mathematica*?

```
Table[{x, x + Sin[2 Pi x]}, {x, 0, 24}]
```

■ 4.2.4. Wechsel des Plot-Stils:
€ Changement du style "plot":

Liniendicke, Graustufe, Farbe, Linienart, etc. können angepasst werden:
• Epaisseur de la ligne, nuance de gris, couleur, type de ligne etc. peuvent être adaptés:

```
Options[Plot]

??PlotStyle

??Graphics

PlotStyle /. Options[Plot]
```

Probiere aus: • Essaie:

```
Plot[x, {x, 0, 10}, PlotStyle -> Thickness[0.125]];

Plot[x, {x, 0, 10}, PlotStyle -> Thickness[0.03]];

Plot[x, {x, 0, 10}, PlotStyle -> RGBColor[0.8,
                                  0.2, 0.2]];

Plot[x, {x, 0, 10}, PlotStyle -> GrayLevel[0.5]];

Plot[x, {x, 0, 10}, PlotStyle -> Dashing[{0.04}]];

Plot[Sin[x], {x, 0, 10}, PlotStyle -> {{Thickness[
       0.02], GrayLevel[0.7], Dashing[{0.1}]}}];
```

■ 4.3. Mehrere Graphen in einem Bild
€ Plusieurs graphes en un image

Man kann auch eine Menge von Funktionen auf einmal ausgeben:
• On peut aussi sortir un ensemble de fonctions à la fois:

```
Plot[{E^x, x^E}, {x, 0, 5}];
```

Da es manchmal schwierig ist, die Kurven zu unterscheiden, ist es angebracht, diese verschieden darzustellen:
• Comme il est parfois difficile de distinquer les courbes, il est préférable de les représenter de differentes façons:

```
Plot[{E^x, x^E}, {x, 0, 5}, PlotStyle -> {
       {Thickness[0.02], Dashing[{0.05, 0.03}]},
       {Thickness[0.01]}     }            ];

Plot[Evaluate[Table[x^(1/n), {n,5}]], {x, 0, 5}];

Evaluate[Table[x^(1/n), {n,5}]]

Table[x^(1/n), {n,5}]
```

"Evaluate" steuert die Operationshierarchie. Bevor die Plotpunkte berechnet werden können, muss die Tabelle generiert werden! Versuche Dich nun in eigenen Plots!

• "Evaluate" dirige la hiérarchie des opérations. Avant de pouvoir calculer les points "plot", il faut générer le tableau! Essaie des plots à toi!

■ Parametrisierte Kurven
€ Courbes paramétrisées

In einem kartesischen Koordinatensystem in der Ebene kann eine Kurve durch einen Vektor gegeben werden, der von einer Variable (Parameter) abhängt. Beispiel:
• Dans un système de coordonnés cartésien, dans le plan, une courbe peut être donnée par un vecteur qui dépend d'une variable (paramètre). Exemple:

```
v[t_]:= {4 Cos[-11 t/4] + 7 Cos[t], 4 Sin[-11 t/4]
          + 7 Sin[t]}; v[t]

ParametricPlot[v[t], {t, 0, 8 Pi}];

ParametricPlot[{4 Cos[-11 t/4] + 7 Cos[t],
                4 Sin[-11 t/4] + 7 Sin[t]},
                {t, 0, 8 Pi}];

ParametricPlot[{4 Cos[-11 t/4] + 7 Cos[t],
                4 Sin[-11 t/4] + 7 Sin[t]},
                {t, 0, 8 Pi}, Axes -> None];
```

Das Achsenverhältnis:
• Rapport axial:

```
AspectRatio /. Options[Plot]

ParametricPlot[{4 Cos[-11 t/4] + 7 Cos[t],
                4 Sin[-11 t/4] + 7 Sin[t]},
                {t, 0, 8 Pi}, Axes -> None,
                AspectRatio -> Automatic];

ParametricPlot[{Cos[t], Sin[t]}, {t, 0, 2 Pi},
                AspectRatio -> 1];
```

Man beachte den Massstab:
• Tenir compte de l'échelle:

```
ParametricPlot[{Cos[t], 5 Sin[t]}, {t, 0, 2 Pi},
                AspectRatio -> 1];

ParametricPlot[{Cos[t], 5 Sin[t]}, {t, 0, 2 Pi},
                AspectRatio -> 5];

ParametricPlot[{Cos[t], 5 Sin[t]}, {t, 0, 2 Pi},
                AspectRatio -> 1/5];
```

■ 4.5. Weitere Optionen
€ Autres options

Z.B. ein Gitter:
• P. ex. une grille:

```
Plot[E^-x^2 Cos[20x], {x,-2,2},
          GridLines -> Automatic];
```

Oder ein Rahmen:
• Ou un cadre:

```
Plot[E^-x^2 Cos[20x], {x,-2,2}, Frame -> True];

Plot[BesselJ[0, x], {x, 0, 20}, Frame -> True];
```

■ 4.6. Plots von Flächen im Raum
€ Plots de surface dans l'espace

Probiere aus! (Hier werden die Achsen beschriftet):
• Essaie! (Ici on met une inscription sur les axes:)

```
Plot3D[Sin[x y],{x, 0, Pi}, {y, 0, 2 Pi},
          AxesLabel -> {"x", "y", "z"}];

Show[%, ViewPoint->{1.050, 1.910, 2.440}];
```

Im Menu unter ("Action", "Prepare Input") neu: "Input", "3D ViewPoint Selector" findest Du eine Hilfe, um den Blickpunkt zu verändern. Du kannst da schieben und dann z.B. die Insert-Taste (Copy-Paste) betätigen, nachdem allerdings der Cursor in einer neuen Zelle parkiert worden ist. Du bekommst dann ein lauffähiges Programm. Probiere die Sache aus!
• Tu trouves dans le menu sous ("Action", "Prepare Input") nouveau: "Input", "3D ViewPoint Selector" une aide pour changer le point de vue. Là tu peux déplacer et ensuite p.ex. appuyer sur la touche insert, après avoir note bien parqué le curseur dans une nouvelle cellule. Tu obtiendras un programme qui fonctionne. Essaie-le!

■ 4.7. Options für Farben, Beleuchtung, ...
€ Options pour couleurs, éclairage, ...

Probiere aus: • Essaie!

```
Options[Plot3D]

LightSources /. Options[Plot3D]

Lighting /. Options[Plot3D]
```

Probiere aus: • Essaie!

```
??SetOptions

SetOptions[Plot3D, Lighting -> False]

Plot3D[Sin[x y],{x, 0, Pi}, {y, 0, 2 Pi}];
```

Oder auf einmal:
• Ou à la fois:


```
Plot3D[Sin[x y],{x, 0, Pi}, {y, 0, 2 Pi},
Lighting -> False];
Plot3D[Sin[x y],{x, 0, Pi}, {y, 0, 2 Pi},
Lighting -> True];
```

Probiere die Options ein wenig aus!
• Essaie un peu les options!

```
?ClipFill

Plot3D[1/Sin[x y],{x, 0.1, Pi-0.1}, {y, 0.1, 2 Pi},
ClipFill -> None];

Plot3D[1/Sin[x y],{x, 0.1, Pi-0.1}, {y, 0.1, 2 Pi},
BoxRatios -> {1,2,4}];
```

■ 4.8. Arbeiten mit Daten
€ Travailler avec des données

■ 4.8.1. Allgemeines
€ Notions générales

Z.B. mit "ListPlot" und "ListPlot3D"
• P.ex. avec "ListPlot" et "ListPlot3D"

```
??ListPlot

??ListPlot3D
```

Erst Daten generieren:
• D'abort générer des données:

```
data = N[Table[{x, Sin[x] + x + 0.5 Random[]},
                {x, 0, 5 Pi, Pi/8}]]
```

Dann Daten ploten:
• Ensuite "ploter" les données:

```
dataplot = ListPlot[data];
```

Polynom durch diese Punkte als Funktion von x:
Polynôme à travers ces points comme une fonction de x:

```
??InterpolatingPolynomial

InterpolatingPolynomial[data,x]

Chop[InterpolatingPolynomial[data,x]]

interplot = Plot[Evaluate[
    InterpolatingPolynomial[data,x]],{x,3,12}];

Plot[Evaluate[
    InterpolatingPolynomial[data,x]],{x,0,20}];
```

```
Show[dataplot, interplot];
```

■ 4.8.2. "Regressionskurven" € "Courbes de régression"

Arbeiten mit "Fit":
• Travailler avec "Fit":

```
??Fit
```

Gerade durch die Punktwolke:
• En ligne droite à travers le nuage de points:

```
trent = Fit[data, {1, x}, x]

trentplot1 = Plot[trent,{x,0,15}];

Show[trentplot1, dataplot];
```

Polynomkurve 6. Grades durch die Punktwolke:
• Courbe polynomiale de 6e degré à travers le huage de points:

```
trent6 = Fit[data, {1, x, x^2, x^3, x^4,
                    x^5, x^6}, x]

trentplot6 = Plot[trent6,{x,0,15}];

Show[trentplot6, dataplot];
```

Kurve als Ueberlagerung von Gerade und Cosinus- sowie
Sinusfunktion durch die Punktwolke:
• Courbe comme superposition d'une droite et d'une fonction cosinus ainsi que sinus à travers le nuage de points:

```
trentcc = Fit[data, {1, x, Cos[x], Sin[x]}, x]

trentplotcc = Plot[trentcc,{x,0,15}];

Show[trentplotcc, trentplot, dataplot];
```

Optionen: • Options:

```
dataplot1 = ListPlot[data, PlotStyle->{PointSize[0.02],
                           GrayLevel[0.2]}];
```

Punkte verbinden:
• Relier les points:

```
dataplot2 = ListPlot[data,  PlotJoined->True];
```

ListPlot kann auf einmal nur eine Datenmenge ploten, jedoch kann man solche Plots überlagern:
• ListPlot ne peut "ploter" qu'un ensemble de données à la fois, on peut cependant superposer de tels plots:


```
data1 = N[Table[{x, 1 + Cos[x] + 0.5 x +
    0.5 Random[]}, {x, 0, 5 Pi, Pi/8}]];
dataplot3 = ListPlot[data1,
    PlotStyle->{PointSize[0.014]}];

Show[dataplot1,dataplot3];
```

■ 4.8.3. Unterdrückung des Graphik-Outputs (Zeit sparen!) € Supprimer la sortie des graphes (économie de temps!)

Studiere: Zuerst Daten generieren (Tabelle mit 3 Listen zu je 5 Elementen, "DisplayFunction"):
• Etudie: D'abort générer les données (tableau à 3 listes à 5 éléments, "DisplayFunction"):

```
myMatrix = Table[Random[], {3}, {5}]

??DisplayFunction
```

Graphiken rechnen, jedoch nicht einzeln ausgeben:
• Calculer des graphes, cependant ne pas les sortir:

```
plot1 = ListPlot[myMatrix[[1]],
    PlotStyle->RGBColor[1,0,0], PlotJoined->True,
    DisplayFunction->Identity];
plot2 = ListPlot[myMatrix[[2]],
    PlotStyle->RGBColor[0,1,0], PlotJoined->True,
    DisplayFunction->Identity];
plot3 = ListPlot[myMatrix[[3]],
    PlotStyle->RGBColor[0,0,1], PlotJoined->True,
    DisplayFunction->Identity];
```

Graphiken ausgeben, alle zusammen in einem Bild:
• Sortir les graphes, tous ensemble en une image:

```
Show[plot1, plot2, plot3,
    DisplayFunction :> $DisplayFunction];
```

Erklärungen: • Explications:

```
??$DisplayFunction

Show[plot1, plot2, plot3,
    DisplayFunction -> $DisplayFunction];

??->

??:>
```

■ 4.8.4. Plot von 3-dimensionalen Daten € Plot des données à 3 dimensions

Erklärungen: • Explications:

```
??ListContourPlot

??ListDensityPlot
```

Erst Daten generieren:
• D'abort générer des données:

```
array1 = Table[x + Sin[x] + Random[], {6},
    {x, 1, 6}]; Print[array1];
                ListPlot3D[array1];

array2 = Table[x + Sin[x] + Random[], {10},
    {x, 1, 10}];    ListPlot3D[array2];
```

Zur besseren Sichtbarmachung zeichnet *Mathematica* eine Fläche und nicht eine Punktwolke im 3-dimensionalen Raum.
ListContourPlot zeichnet davon eine
Höhenlinienkarte:
• Pour une meilleure perceptibilité, *Mathematica* dessine une surface et non pas un nuage de points dans l'espace à trois dimensions. ListContourPlot dessine une carte indiquant les courbes à niveau:

```
ListContourPlot[array1];
```

Eine andere Möglichkeit zur Veranschaulichung bietet ListDensityPlot:
ListDensityPlot offre une autre possibilité d'illustration:

```
ListDensityPlot[array1];
```

■ 4.9. Eingebaute Graphikelemente oder Blǒcke € Eléments ou plocs graphiques incorporés

■ 4.9.1. Allgemeines € Notions générales

Mathematica bietet die Möglichkeit, Linien, Punkte, Kreise, Scheiben und Polygone direkt zu zeichnen. Beispiele:
• *Mathematica* offre la possibilité de dessigner directement des lignes, points, cercles, disques, polygones. Exemples:

```
Show[Graphics[{Circle[{0,0},.1]}]];
```

Wieso ist aus dem Kreis eine Ellipse geworden?
• Pourquio le cercle est-il devenu une ellipse?

```
Options[Graphics][[1]]
```

Man muss also das Achsenverhältnis anpassen!
• Il faut donc adapter le rapport des axes:

```
Show[Graphics[{Circle[{0,0},.1]},
    AspectRatio->1]];

Show[Graphics[{Circle[{0,0},.1]},
    AspectRatio->Automatic]];

Show[Graphics[{Circle[{0,0},.1]},
    AspectRatio->2]];
```

Oder: • Ou:


```
Show[Graphics[{
    PointSize[0.05], Point[{0, 0}], Point[{1, 0}],
    Line[{{0, 0},{1, 0}}],
    Circle[{0,0},.1], Circle[{1,0},.1]},
    AspectRatio->Automatic]];
```

Oder ein Punkt im 3-dimensionalen Raum:
• Ou un point dans l'espace à trois dimensions:

```
Show[Graphics3D[Point[{3,2,1}]]];

Show[Graphics3D[{PointSize[0.05],
        Point[{3,2,1}]}]];
```

Oder erst Liste generieren:
• Ou générer d'abort une liste:

```
pts = Table[Random[], {4}, {3}]
```

Dann Liste manipulieren:
• Ensuite manipuler la liste:

```
??Map

graphicPts = Map[Point, pts]
```

Dann davon Plot machen:
• Ensuite faire un plot:

```
Show[Graphics3D[{PointSize[0.03], graphicPts}]];
```

Mit Skala: • Avec échelle:

```
Show[Graphics3D[{PointSize[0.03], graphicPts}],
                Axes->Automatic];
```

■ 4.9.2. Umwandlung von graphischen Objekten: € Transformation d'objets graphiques:

Mit Hilfe welcher Graphic-Objekte wird ein Plot ausgeführt?
• A l'aide de quels objets graphiques exécute-t-on un plot?

```
??InputForm

basicPlot = Plot[x^2, {x, 0, 25}];

InputForm[basicPlot]
```

Trick: Ausgabe mehrerer Plots auf einmal:
• Truc: Emission de plusieurs plots à la fois:

```
Show[
    Plot[Sin[x], {x, 0, 3 Pi}],
    Graphics[{PointSize[0.05],
    Table[Point[{n Pi, 0}], {n, 0, 3}]
    }]
    ];
```

Hier sind 2 Plots gekommen. Damit nur der letzte kommt, muss man mit "DisplayFunction" arbeiten:
• Voici 2 plots sortis. Pour faire sortir seulement le dernier plot, il faut travailler avec "DisplayFunction":

```
Show[
    Plot[Sin[x], {x, 0, 3 Pi},
        DisplayFunction->Identity],
        Graphics[{PointSize[0.05],
        Table[Point[{n Pi, 0}], {n, 0, 3}]
        }], DisplayFunction->$DisplayFunction
        ];
```

■ 4.10. Beschriftung von Graphiken (Labels)
€ Doter des graphiques d'inscriptions (Labels)

Beschriftung der x-Achse. Zuerst ohne:
• Faire des inscriptions sur l'axe x. D'abort sans inscriptions:

```
Plot[Sin[x], {x, 0, 3 Pi}];
```

Dann mit: • Ensuite avec inscription:

```
??Ticks
```

```
Plot3D[Sin[x y], {x, 0, 3 Pi}, {y, 0, Pi}];
```

Labels: • Labels:

```
??PlotLabel
```

```
??FontForm
```

\n: Neue Linie
• \n: Nouvelle ligne:

```
Plot3D[Sin[x y], {x, 0, 3 Pi}, {y, 0, Pi},
    PlotLabel -> FontForm["Hallo friend!
    Plot3D[Sin[x y], \n
    {x, 0, 3 Pi}, {y, 0, Pi}]",
    {"Courier-Bold", 15}]];
```

Oder: • Ou:

```
??Text
```

```
Show[Graphics[{Circle[{0,0},1],
    Text["Circle",{0,0.05}]}],
    AspectRatio -> Automatic];
```


■ 4.11. Graphik-Pakete
€ Paquets de graphiques

■ **Zum Beispiel Rotationskörper:**
€ Par exemple corps de rotation:

Package laden: • Charger Package:

```
<< Graphics/Shapes.m
```

Beispiele: • Exemples:

```
Show[Graphics3D[Torus[]]];
```

```
Show[RotateShape[Graphics3D[Torus[]],
        0, 5 Pi/6, 0]];
```

■ 4.12. Animationen
€ Animations

■ **Auf dem Buckdeckel von Nancy Blachmans Buch:**
€ De la couverture du livre de Nancy Blachman:

Zuerst Graphiken rechnen (hier nur wenige Bilder zur schnellen und kurzen Demonstration, Bilderzahl bitte im Praktikum selbst erhöhen):
• D'abort calculer des graphiques (ici quelques images pour und démonstration rapide e brève, augmenter soi-même le nombre d'images lors du travail pratique):

```
cover = Table[Plot3D[Sin[x y], {x, 0, t/2},
        {y, 0, t}, PlotPoints -> 9,
        Ticks -> None],
         {t, 1, 6, 3/4}]
```

■ **Vorgehen zum Animieren:**
€ Procédé d'animation:

Jede der eben ausgegebenen Graphiken ist in einer eigenen Zelle untergebracht.
Alle diese Graphikzellen wiederum bilden eine gemeinsame Zelle (vgl. von links aus 2. senkrechte Linie am rechten Rand). Diese Zelle ist mit der Maus anzuklicken. (Sie wird dann dunkel markiert.) Nun wählst Du mit der Maus im Menu ("Graph") neu "Cell" und dann "Animate Selected Graphics ....y".
Du wirst sehen, was geschieht!
Um die Sache anzuhalten, brauchst Du nur mit der Maus irgendwohin ausserhalb der eben gewählten Menuzeile zu "klicken".
• Chacun des graphiques qui viennent de sortir est placé dans une cellule à lui. Toutes ces cellules de graphiques constituent de nouveau une cellule en commun (compare de gauche la 2e ligne verticale sur le bord de droite). Il faut toucher cette ligne avec le souris. (Elle deviendra foncée.) Maintenant tu choisis avec le souris dans le menu ("Graph") nouveau "Cell" et ensuite "Animate Selected Graphics ....y". Regarde ce qui se passe! Pour arrêter le processus, il te suffit d'activer la souris n'importe où en dehors du menue que tu viens de choisir.

■ **Eine verkleinerte Darstellung:**
€ Une représentation rapetissée:

```
??GraphicsArray

??GraphicsSpacing

Show[GraphicsArray[cover],
GraphicsSpacing -> 0.2];
```

■ 4. 13. PostScript
€ PostScript

■ **Erzeugen, ablegen, einlesen und betrachten von PostScript-Code**
€ Générer, déposer, lire et observer le code PostScript

Einige Erklärungen zu Befehlen:
• Quelques explications quant aux ordres:

```
??>>

??Display

??<<
```

Wir erzeugen eine Graphik:
• Nous générons un graphique:

```
expCos = Plot[Exp[-x^2] Cos[20 x], {x, -2, 2}];
exp1   = Plot[Exp[-x^2], {x, -2, 2}];
```

Nun schauen wir uns den Inhalt des Files "postscript_test1.ps" auf dem Schirm an. Du siehst sofort, dass hier offenbar Koordinaten von gerechneten Punkten gespeichert worden sind. Arbeite dazu die folgenden Befehle ab:

• Maintenant nous regardons le contenu des fichiers (files) "postscript_test1.ps" sur l'écran. Tu vois tout de suite qu'ici on a évidemment sauvé les coordonnés des points calculés. Exécute les ordres suivants:

```
?exp1
```

Nun schreiben wir mit ">>" den PostScript-Code in ein File namens "postscript_test.ps" u.s.w. ins eigene Unterverzeichnis: (Hinweis: Der Name des Unterverzeichnisses ist hier "c:\work".)
• Maintenant nous écrivons avec ">>" le code "PostScript" dans un fichier du nom de "postscript_test.ps" etc. dans notre propre sous-dossier: (Indication: Le nom du dossier qu'on a ici est "c:\work".)

```
expCos >> "c:\work\postscri.ps";
exp1 >> "c:\work\postsc_1.ps";
```

Führe den Befehl aus und kontrolliere rasch im eigenen Verzeichnis, ob das File jetzt da steht! (Vielleicht vergeht etwas Zeit, bis es kommt.)
"Display" tut dasselbe. Statt in ein PostScript-File speichern wir nachher den Code mit "Save" in ein Mathematica- File, das wir später wieder einlesen und den Mathematica-Code verwenden können. (Ausführen und kontrollieren! Probiere im einzulesenden File erst den PostScript-Code ein wenig zu verändern. Hüte Dich dabei vor Werten, die nicht verarbeitet werden können!):
• Execute l'ordre et contrôle vite dans le propre dossier, si le fichier y est! (Peut-être tu dois attendre un peu qu'il apparaisse.) "Display" fait la même chose. Au lieu de sauver le code dans un fichier PostScript, nous le sauvons après avec "Save" dans un fichier de *Mathematica*, que nous lisons plus tard en utilisant le code de *Mathematica*. (Faire et contrôler! Essaie de changer d'abord un peu le code PostScript dans le fichier à lire. Evite les valeurs qui ne peuvent pas être élaborées!)

```
Display["c:\work\postsc_2.ps",expCos];
Display["c:\work\postsc_3.ps",exp1]

Save[
  "c:\work\postscri.ma",expCos];
Save[
  "c:\work\postsc_1.ma",exp1]

Save[
  "c:\work\postscri.nb",expCos];
Save[
  "c:\work\postsc_1.nb",exp1]
```

Nun löschen wir den PostScript-Code "expCos" u.s.w.:
• Maintenant nous effaçons le code PostScript "expCos" etc.:

```
Remove[expCos, exp1]

??exp1

??expCos
```

Das File kann wieder eingelesen werden:
• On peut relire le fichier:

```
??Get

Get["c:\work\postsc_1.nb",exp1]

?exp1
```

```
Show[Graphics[expCos]];
Show[Graphics[exp1]];
```

Der Befehl ist jetzt wieder da! Nun schauen wir uns den Inhalt des Files "postscri.ps" auf dem Schirm an.
• Voici de nouveau l'ordre! Maintenant nous regardons le contenu du fichier "postscri.ps" sur l'écran.

```
!!"c:\work\postscri.ps"

Get["c:\work\postscri.ps"]

Get["c:\work\postscri.ma"]

Get["c:\work\postscri.nb"]

exp1
```

■ 4. 14. Experimentieren mit Animationen
€ Expérimenter les animations

■ 4.14.1. Nachfolgend zwei Beispiele zu Animationen (aus einem File von U. Altmann)
€ Ensuite deux exemples d'animations (pris d'un fichier de U.Altmann)

Achtung! Die Rechnungen dauern vielleicht etwas Zeit!
Vorschlag: Erst am Schluss der Arbeit ausführen!
€
Attention! Il faut du temps aux calculs! Proposition:
Executer à la fin du travail!

Package laden:
• Charger Package:

```
Needs["Graphics`ImplicitPlot`"]
```

Beispiel: Zuerst n = 128 Graphiken rechnen (das dauert!!!). Um Ueberraschungen zu vermeiden, ist n auf 10 eingestellt. Das geht zwar schneller, aber die Animation wird so nicht gut. Sonst kann es schon eine Lektion dauern!
• Exemple: Calculer d'abort n = 128 graphiques (cela dure!!!). Pour éviter des surprises, il faut mettre n sur 10. Cela va plus vite, mais l'animation n'est pas si bonne. Si non cela peut durer tout une leçon.


```
a=3;
(* n=128; *)
n=10;
tabelle1=Table[
  ImplicitPlot[(x-y+2)(x^2+y^2-1) ==
    N[3.(1.-Tan[b])] x, {x,-a,a},
    PlotRange->{{-a,a},{-a,a}}, AspectRatio->1.],
    {b,-Pi/2+Pi/n,Pi/2-Pi/n,Pi/n}];
Prepend[tabelle1,
  ParametricPlot[{0.,t},{t,-a,a},
    PlotRange->{{-a,a},{-a,a}}, AspectRatio->1.]];
```

Dann animieren!
• Ensuite animer!

2. Beispiel:
Zuerst n = 128 Graphiken rechnen. (Um Ueberraschungen zu vermeiden, ist n auf 10 eingestellt. Das geht zwar schneller, aber die Animation wird so nicht gut.)
• 2e exemple:
D'abord calculer n = 128 graphiques. (Pour éviter des surprises, il faut mettre n sur 10. Cela va plus vite, mais l'animation n'est pas si bonne.)

```
a=3;
(* n=128; *)
n=10;
tabelle2=Table[
  ImplicitPlot[(x-y)(x^2+y^2-1)==N[Tan[b]-1] x,
    {x,-a,a}, PlotRange->{{-a,a},{-a,a}},
    AspectRatio->1.],{b,-Pi/2+Pi/n,Pi/2-Pi/n,Pi/n}];
Prepend[tabelle2,
  ParametricPlot[{0.,t},{t,-a,a}, PlotRange->{{-a,a},
  {-a,a}},AspectRatio->1.]];
```

Dann animieren!
• Ensuite animer!

■ 4.14.2. Noch ein Beispiel (Simulation einer Welle)
€ Encore un exemple (simulation d'une onde)

■ Erst rechnen, dann animieren!
€ D'abord calculer, ensuite animer!)

(Um Ueberraschungen zu vermeiden, sind statt 60 nur 5 Schritte eingestellt.
Das geht zwar schneller, aber die Animation wird so nicht gut.)
• (Pour éviter des surprises, il faut 5 au lieu de 60 étapes. Cela va plus vite, mais l'animation réussit moins bien.)

```
Table[Plot3D[
    Sin[3t+(x-3)^2+(y-2)^2]/(1+(x-3)^2+(y-2)^2)+
    3/5Sin[5/3(3t+(x+3)^2+(y-2)^2)]/(1+(x+3)^2+(y-2)^2)+
    3/4Sin[4/3(3t+x^2+(y+3)^2)]/(1+x^2+(y+3)^2),
    {x,-8,8},{y,-8,8},
PlotRange->{{-8,8},{-8,8},{-1,1}},
PlotPoints->30],
(* {t,0,2Pi,2Pi/60}];  *)
{t,0,2Pi,2Pi/5}];
```

Wähle die Zelle an, die alle ausgegebenen Bilder umschliesst. Animiere mit "Animate Selected Graphics ...y" in Menu ("Graph") neu "Cell". Beobachte, welche Zonen sich bewegen und welche Zonen ruhig bleiben. Gibt es Interferenzen?

• Choisir la cellule qui inclut toutes les images sortis. Anime par "Animate Selected Graphics ...y" dans le menu ("Graph") nouveau "Cell". Observe quelles zones restent tranquilles. Y a-t-il des interférences?

■ 4.15. Probiere eigene Beispiele aus!
€ Essaie de propres exemples!

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Uebungen € Exercices

4. Graphiken
€ Graphiques

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 4 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 4.
WIR94/99

■ Aufgabe 1 € Problème 1

■ Spiele ein wenig mit den Options
€ Joue un peu avec les options

Sinus ohne ";"

• Sinus sans ";"

```
Plot[Sin[x],{x,0,6 Pi},
    AspectRatio -> 1/2,
    PlotLabel -> "Sin[x]",
    AxesLabel -> {"x", "y"},
    AxesOrigin -> {2 Pi, 1/2} ]
```

Sinus mit ";" --- Unterschied?

• Sinus avec ";" --- y a-t-il une différence?

```
Plot[Sin[x],{x,0,6 Pi},
    AspectRatio -> 1/2,
    PlotLabel -> "Sin[x]",
    AxesLabel -> {"x", "y"},
    AxesOrigin -> {2 Pi, 1/2} ];
```

■ Aufgabe 2 € Problème 2

■ (a) Nullstellen schätzen
€ Evaluer les zéros (estimer...)

Schätze die Nullstellen mit Hilfe des Plots:
• Evalue les zéros à l'aide de "Plots":

```
Plot[2x^3-7x^2-17x+10,{x,-6,6}];
```

Verifiziere die Schätzung mit "Solve":
Vérifiér l'évaluation par "Solve":

```
Solve[2x^3-7x^2-17x+10==0,x]
```

■ (a) Nullstellen schätzen aus Graph:
€ Evaluer les zéros d'un graphe:

Schätze die Nullstellen mit Hilfe des Plots auf 3 Stellen (herauszoomen):
• Evalue les zéros à l'aide des "Plots" (3 places, zoom):

```
Plot[BesselJ[0,x],{x,-6,6}];
```

Verifiziere die Schätzung mit "Solve":
• Vérifie l'évaluation pour "Solve":

```
Solve[2x^3-7x^2-17x+10==0,x]
```

■ Aufgabe 3 € Problème 3

■ Verschiedene Dinge in einem Graphen:
€ Plusieurs choses dans un graphe:

Zeichne den Graphen der Besselfunktion J0 sowie Liniensegmente, die vom Startpunkt der Newton-Iteration für die Nullstellen r Nullstelle führen:
• Dessine le graphe de la fonction d'après Bessel J0 ainsi que des segments de lignes, qui vont du point de départ de l'itération d'après Newton pour les zéros au zéro:


```
Show[
    Plot[BesselJ[0,x],{x,0,20},
        DisplayFunction -> Identity],
    Table[Graphics[{Dashing[{1/(10t)}],
                    Thickness[0.01],
                    Line[{
                        {t,BesselJ[0,t]},
                        {x/. FindRoot[BesselJ[0,x]==0,
                                      {x,t}],0}
                        }]
                    }],
                    {t,0,20,3}
                ],
                DisplayFunction :> $DisplayFunction
                ];
```

■ Aufgabe 4 € Problème 4

■ (a) Plot einer Funktion und ihrer Potenzreihe:
€ Plot d'une fonction et sa série de puissances:

f definieren:
• Définir f:

```
Clear[f]; f[x_]:=1/(1 + Sin[Sqrt[Pi^2 + x]^2]); f[x]
```

Verwende Series und Normal: Achtung: Viel Output!
• Utilise "Series" et "Normal": Attention: Beaucoup d'Output!

```
n[x_]:=Normal[Series[f[x],{x,0,8}]];
nTest[x_]:=Normal[Series[f[x],{x,0,3}]];
nTest[x]
```

Plot von f und n:
• Plot de f et n:

```
Print[N[n[x]]]

a = 1;
Plot[Evaluate[N[n[x]]],Evaluate[{x,-a,a}],
                PlotStyle->{{Thickness[0.01],
        Dashing[{0.04}]}}];

a = 1; (*PlotRange*)
Show[
    Plot[f[x],Evaluate[{x,-a,a}],
        DisplayFunction -> Identity,
        PlotStyle->{GrayLevel[0.1]}],
    Plot[Evaluate[N[n[x]]],Evaluate[{x,-a,a}],
        DisplayFunction -> Identity,
        PlotStyle->{{Thickness[0.01],
        Dashing[{0.04}]}}],
        DisplayFunction :> $DisplayFunction];
```

■ Aufgabe 5 € Problème 5

■ (a) Probleme mit PlotPoints
€ Problèmes avec Plotpoints

x^2+cos(22x) :
Fehler beim Plotten bei normal vielen PlotPoints (in alten Versionen):
• Erreurs qui apparaissent en plotant avec un nombre normal de Plotpoints (vieilles vérsions):

```
Plot[x^2 + Cos[22x], {x,-5,5}];
```

```
Plot[x^2 + Cos[22 x], {x, -5, 5}, PlotPoints -> 9];
```

■ (b) Korrektur mit PlotPoints

x^2+cos(22x) :
Wähle 70 PlotPoints:
• Choisir 70 "PlotPoints":

■ Aufgabe 6 € Problème 6

■ Einige ParametricPlots:
€ Quelques "ParametricPlots":

■ (a)

```
ParametricPlot[{Cos[4t] Cos[t],Cos[4t] Sin[t]},{t,0,2 Pi},
AspectRatio->Automatic];
```

■ (b)

```
ParametricPlot[{Cos[7t] Cos[3t],Cos[7t] Sin[3t]},{t,0,2 Pi},
AspectRatio->Automatic];
```

■ (c)

```
ParametricPlot[{Cos[7t] Cos[11t],Cos[7t] Sin[11t]},{t,0,2 Pi},
AspectRatio->Automatic];
```


■ Aufgabe 7 € Problème 7

■ Spiel mit dem Package "Polyhedra.m"
€ Jeu avec le Package "Polyhedra.m"

Einbinden: • Lier:

```
Needs["Graphics`Polyhedra`"];
```

Graphik ansehen:
• Regarder le graphique:

```
Show[Graphics3D[Icosahedron[]]];
```

Graphik ansehen:
• Regarder le graphique:

```
Show[Graphics3D[Stellate[Icosahedron[],3]],
    Boxed->False];
```

■ Aufgabe 8 € Problème 8

■ Lighting

Normal: • Normal:

```
Plot3D[Sin[x y],{x,-5,5},{y,-5,5}];
```

Lighting false: • Lighting false:

```
Plot3D[Sin[x y],{x,-5,5},{y,-5,5},Lighting->False];
```

■ Aufgabe 9 € Problème 9

■ ViewPoint

Normal: • Normal:

```
Plot3D[Sin[x y],{x,-5,5},{y,-5,5},Lighting->True,
       ViewPoint->{0,0,3}];
```

Anders: • Autrement:

```
Plot3D[Sin[x y],{x,-5,5},{y,-5,5},Lighting->True,
       ViewPoint->{0,0,10}];
```

Nochmals anders:
• Encore autrement:

```
Plot3D[Sin[x y],{x,-5,5},{y,-5,5},Lighting->True,
       ViewPoint->{0,3,10}];
```

Oder so:
• Ou ainsi:

```
Plot3D[Sin[x y],{x,-5,5},{y,-5,5},Lighting->True,
       ViewPoint->{2,-4,6}];
```

■ Aufgabe 10 € Problème 10

■ Contour- und DensityPlot
€ Contour- et DensityPlot

Contour: • Contour:

```
ContourPlot[Sin[x y],{x,-5,5},{y,-5,5}];
```

Density: • Density:

```
DensityPlot[Sin[x y],{x,-5,5},{y,-5,5}];
```

■ Aufgabe 11 € Problème 11

■ Mit ContourPlot zu Ellipsen:
€ Faire des ellipses avec ContourPlot

Contour: • Contour:

```
ContourPlot[x^2 + 2 y^2,{x,-5,5},{y,-5,5},
AspectRatio->Automatic];
```

■ Aufgabe 12 € Problème 12

■ Spiel mit Listen:
€ Jouer avec des listes

Primzahlen generieren:
• Générer des nombres premiers:

```
t = Table[Prime[n],{n,10}]
```


Plot: • Plot:

```
ListPlot[t,PlotStyle->{PointSize[0.05]}];
```

Punkte verbinden:
• Relier des points:

```
ListPlot[t,PlotStyle->{PointSize[0.05]},
       PlotJoined->True];
```

Fit: • Fit:

```
tfit = Fit[t,{1,x,x^2},x]
```

Fit-Kurve ansehen:
Regarder la courbe Fit:

```
Plot[tfit,{x,0,30}];
```

Kurve zusammen mit Punkten ansehen:
• Regarder la courbe et les points ensemble:

```
Show[
    ListPlot[t,PlotStyle->{PointSize[0.05]},
        DisplayFunction -> Identity],
    Plot[tfit,{x,0,10},DisplayFunction -> Identity], DisplayFunction :>
$DisplayFunction
           ];
```

Abweichungen: • Ecarts:

```
Clear[tfit]; tfit[x_] := Fit[t,{1,x,x^2},x];
te = Table[Prime[n]-tfit[n],{n,10}]
```

Abweichungen im Plot:
• Ecarts dans le Plot:

```
Show[
    ListPlot[t,PlotStyle->{PointSize[0.05]},
        DisplayFunction -> Identity],
    ListPlot[te,PlotStyle->{PointSize[0.035]},
        DisplayFunction -> Identity],
    Plot[tfit[x],{x,0,10},DisplayFunction -> Identity],
    DisplayFunction :> $DisplayFunction
               ];
```

■ Aufgabe 13 € Problème 13

■ Zeichnen mit *Mathematica*
€ Dessiner avec *Mathematica*

Also los (Golden Gate Bridge):
• Allez-y (Golden Gate Bridge):

```
Show[
    Graphics[{ (*Brücke*)
        Circle[{-2,1},1,{3 Pi/2, 2 Pi}],
        Circle[{0,1},1,{Pi, 2 Pi}],
        Circle[{2,1},1,{Pi, 3 Pi/2}],
        Line[{{-2,0},{2,0}}],
        Line[{{-1,-0.5},{-1,1}}],
        Line[{{1,-0.5},{1,1}}]        }],
    Graphics[ (*Wasser*)
        Table[Circle[{x,-0.5},0.1,{Pi, 2Pi}],
            {x, -2, 2, 0.2}]         ],
    Graphics[{ (*Fisch*)
        Circle[{0,-1.2},0.4,{Pi/6, 5 Pi/6}],
        Circle[{0,-0.8},0.4,{7 Pi/6, 11 Pi/6}],
        Point[{-0.2,-0.95}],
        Circle[{-0.3,-1},0.2,{- Pi/4, Pi/4}],
        Line[{{0.35,-1},{0.5,-0.85},
             {0.5,-1.15},{0.35,-1}}]
             }],
        AspectRatio->Automatic,
        PlotRange->All  ];
```

■ Aufgabe 14 € Problème 14

■ Zeichnen mit *Mathematica* € Dessiner avec *Mathematica*

Also los (geometrisches Gesicht):

• Allez-y (visage géométrique):

```
Show[
    Graphics[{
        (*Augen*)
        PointSize[0.04],
        Point[{-0.3,0.1}],
        Point[{0.3,0.1}],
        (*Nase*)
        Disk[{0,0},0.35,{23 Pi/16, 25 Pi/16}],
        (*Mund*)
        Line[{{-0.3,-0.5},{0,-0.6}}],
        Line[{{0,-0.6},{0.3,-0.5}}],
        (*Kopf*)
        Circle[{0,0},1]
         }],
        AspectRatio->Automatic,
        PlotRange->All  ];
```


■ Aufgabe 15 € Problème 15

■ ZZeichnen mit *Mathematica* € Dessiner avec *Mathematica*

Wie wirken Farben?:

• Quels sont les effets des couleurs?

```
Show[
    Graphics[
        Table[{
            RGBColor[0,x/4.,1.-x/4.],
            Point[{x, Cos[x^2] - Sin[x]}]},
            {x,0,4,0.01}
            ]
            ],
            Axes->Automatic
        ];
```

■ "Putzmaschine" einsetzen € Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

5. Etwas Umgang mit Mathematica
€ Savoir un peu manier *Mathematica*

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 5 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 5.
WIR94/99

Achtung: *Mathematica* neu starten, sonst könnte sehr viel Output auf den Schirm kommen.....
€ Attention: Démarrer à nouveau *Mathematica*, sinon il pourrait y sortir beaucoup d'output sur l'écran...

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

■ 5.1. Seiten-Breite setzen
€ Fixer la largeur des pages

■ Ein Vorspann
€ Préparatifs

"Material" erzeugen zur späteren Benutzung:
• Générer des "materiaux" à utiliser plus tard:

```
(*n30*)

Clear[n30]

n30[x_]:=N[x,30]
```



```
a = 4

n30[Pi]

Plot[-2x,{x,-1,1}];
```

Anzahl Output-Charakter verändern:
• Changer le charactère de l'output:

```
??SetOpt*

??$Output

??Pag*

??PageWidth

SetOptions[$Output, PageWidth -> 45]

Options[$Output]
```

In 5.2. werden folgende Befehle besprochen: "In", "Out". Probiere:
• Dans 5.2. on discute les ordres suivants: "In", "Out". Essaie:

```
??In

linie = $Line;
??Out

linie
```

"Material" erzeugen zur späteren Benutzung:
• Générer des "materiaux" à utiliser plus tard:

```
Plot[Sin[x],{x,0,50}];

Plot[x^2,{x,0,1}];
```

■ 5.2. Listings von Input und Output
€ Listings d'input et d'output

■ 5.2.1. Input-Lines auflisten
€ Faire une liste des input-lines

Mit "Recall" in älteren Versionen. Neu mit "InString":
• Dans les versions plus vieilles avec "Recall". Dans les versions nouvelles avec "InString":

```
Recall[1, 2, 3]

??InString

InString[{1, 2, 3, 4, 5, 6}]

In[1]
```

```
Out[1]
```

Achtung: Wenn Du *Mathematica* nicht neu gestartet hast, so könnte hier sehr viel Output auf den Schirm kommen..... Es könnte eine Stunde oder mehr dauern!
€ Attention: Démarrer à nouveau *Mathematica*, sinon il pourrait y sortir beaucoup d'output sur l'écran... Cela pourrait durer plus d'une heure!

```
??In
```

Das Folgende gibt sehr viel Output (zum Drucken hier gespert):
• Ce qui suit produit beaucoup d'output (ici défense d'imprimer):

```
(* ??Out *)
```

■ 5.2.2. In ein File schreiben und wieder abrufen:
€ Ecrire dans un fichier et rappeler:

```
Save["C:\work\ASession.nb",
          In,  linie, (*Out*)]

!!C:\work\ASession.nb
```

■ 5.3. Zum Editor
€ Quant à l'éditeur

■ **Hier für "NeXT"! (Für andere Editoren ev. anders, vgl. *Mathematica*, a Practical Approach von Nancy Blachman oder Literatur)**
€ Ici seulement pour "NeXT"! (Pour d'autres éditeurs voir *Mathematica*, a Practical Approach de Nancy Blachman ou littérature)

Einige Edit-Funktionen:
• Quelques fonctions Edit:

```
??Edit

Edit[expr___] lets you edit the expressions expr.

??EditDefinition

??EditDef

??EditIn

??*Edit*
```


Beispiele: • Exemples

```
Edit[Expand[(x+2y)^6]]
```

Oder: • Ou:

```
EditDefinition[n30]

EditDefinition[a]
```

Funktioniert der Befehl auf NeXT?
• L'ordre fonctionne-t-il sur NeXT?

```
??n30

??a
```

■ 5.4. Zu On-Line Help
€ Quant à On-Line Help

■ Ein Trick
€ Un truc

Weisst Du beim Schreiben innerhalb eines Befehls nicht mehr weiter, so kannst Du in der selben Zelle die Help-Funktion benutzen.
• Si tu ne t'en sors plus en écrivant à l'intérieur d'un ordre, tu peux utiliser la fonction Help dans la même cellule.

Beispiel: • Exemple:

```
Series[Csc[x],
?Series
```

Natürlich kannst Du auch eine neue Zelle machen:
• Naturellement tu peux faire une nouvelle cellule:

```
?Csc
```

■ 5.5. Ein Blank kann "oder" bedeuten...
€ Un "blanc" peut signifier "ou":

Das Problem besteht nur bei Version 1.2.:
• Le problème existe seulement pour la version 1.2.:

```
?X* Z*
```

■ 5.6. Symbollisten erzeugen:
€ Générer des listes de symboles

■ 5.6.1. Ein Beispiel, das alles sagt:
€ Un exemple qui dit tout:

```
Names["W*"]
```

Oder: • Ou:

```
Names["A*"]
```

Das Problem besteht nur bei Version 1.2.:
• Le problème existe seulement pour la version 1.2.:

■ 5.6.2. Jetzt ganz *Mathematica* sowie alle bisherigen eigenen Definitionen auf einmal:
€ Maintenant tout *Mathematica* ainsi que toutes les propres définitions jusqu'à présent à la fois:

```
Names["*"]
```

■ 5.7. Verschiedene Möglichkeiten, eine Funktion zu codieren:
€ Différentes manières de coder une fonction:

■ 5.7.1. Die Definition einer Funktion:
€ La définition d'une fonction:

Beispiel: Die Funktion "f(x) = sin(x) * cos(2x)" soll definiert werden.
Der Unterstrich "_" gibt die Funktionsvariable an. ":=" bedeutet "definiere":
• Exemple: Il faut définir la fonction "f(x) = sin(x) * cos(2x)". Le trait "_" indique la variable de la fonction. ":=" signifie "définis":

```
f[x_]:= Sin[x] Cos[2 x]
```

Ausgabe zur Kontrolle:
• Sortie de contrôle:

```
f[x]
```

Anwendung: • Application:

```
f[Pi]
```

```
Plot[f[x],{x, 0, 10}];
```


Kurz in einer Zelle eine andere Funktion:
• Brièvement une autre fonction dans une cellule:

```
f1[x_]:= Sin[2x] Cos[3 x^2];
         Plot[f1[x],{x, 0, 10}];
```

Zum Vergleich, wie es nicht funktioniert:
• En tout que comparaison: Voici comme ça ne fonctionne pas:

```
g[x]:= Sin[2x] Cos[3 x^2];
         Plot[g[x],{x, 0, 10}];

g1[x_]= Sin[2x] Cos[3 x^2];
         Plot[g1[x],{x, 0, 10}];

g2[x]= Sin[2x] Cos[3 x^2];
         Plot[g2[x],{x, 0, 10}];
```

■ 5.7.2. Die Prefix-Notation:
€ La notation préfixe:

Beispiel: Die Funktion "N" mit Klammern "[...]":
• Exemple: La fonction "N" avec des parenthèses "[...]":

```
N[Sqrt[2]/2]
```

■ 5.7.3. Die Postfix-Notation:
€ La notation postfixe:

Beispiel: "N" kommt hinten, abgetrennt durch "//":
• Exemple: "N" se place derrière, séparé par "//":

```
Sqrt[2]/2 // N
```

Damit lassen sich "Pipes" bilden. d.h. mehrere Funktionen nacheinander ausführen. Beispiele:
• Ainsi on peut former des "pipes", c'est-à-dire générer plusieurs fonctions l'une après l'autre. Exemples:

```
"W*" // Names
```

```
"W*" // Names // Length
```

```
x = Pi/2
```

```
Sin[x]
```

```
Sin[x] // Sin // N
```

```
Sin[x] // Sin // Sin // N
```

```
Sin[x] // Sin // Sin // Sin // Sin // N
```

```
N[Sin[Sin[Sin[Sin[Sin[x]]]]]]
```

■ 5.8. Output unterdrücken
€ Supprimer l'output

■ 5.8.1. Mit ";" am Schluss
€ Mettre ";" à la fin

Beispiel: • Exemple:

```
Sin[4.08]
```

```
Sin[3.96];
```

Abrufen: • Appeler:

```
%
```

■ 5.8.2. Diverse Befehle in einer Zelle. Welcher Output kommt?
€ Différents ordres dans une cellule. Quel sera l'output?

Beispiele: • Exemples:

```
a=1; b=2; c=3; d=4
```

```
a=1; b=2; c=3; d=4;
```

```
a=1; b=2; Print[b]; c=3; d=4;
```

```
a=1; b=2; Print[b]; c=3; d=4
```

```
a=1; b=2; Plot[x,{x,0,1}]; c=3; d=4
```

■ 5.9. Timing - Rechenzeit messen
€ Timing - mesurer le temps de calcul

Mit "Timing" wird die CPU-Zeit gemessen. Beispiele:
• Avec "Timing" on mesure le temps CPU. Exemple:

```
Timing[N[Pi,500]]
```

Ohne den Output der Rechnung:
• Sans l'output du calcul:

```
Timing[N[Pi,500];]
```

```
Timing[N[Pi,2000];]
```

```
Plot[Sin[x],{x, 0, 50}] // Timing
```

```
Plot[Sin[x],{x, 0, 50}]; // Timing
```


■ 5.10 Abbruch und Unterbruch
€ Rupture et interruption

Während der Rechnungsausführung hast Du im Menu unter ("Action") neu "Kernel" - "Interrupt" die Möglichkeit, mit "Abort" abzubrechen oder mit "Interrupt Calculation" zu unterbrechen.
Der Abbruch ist auf NeXT auch möglich über die Tastatur mit "Command" und "." zusammen gedrückt. U.s.w..
Probiere aus (z.B. 5000 Fakultät....):
• Dans le menu sous ("Action") nouveau "Kernel" - "Interrupt" tu as la possibilité d'interrompre le calcul par "Interrupt Calculation" ou définitivement par "Abort".
Sur NeXT l'interruption définitive est aussi possible avec les touches "Command" et ".", pressées simultanément. Etc..
Essaie (p.ex. 5000 factoriel)

```
5000!
```

Braucht wohl die Bildschirmausgabe oder die Rechnung mehr Zeit? Probiere aus!
• Est-ce la sortie ou le calcul qui dure plus de temps? Essaie!

```
1000! // Timing
```

■ 5.11 Globale Variablen
€ Variables globales

■ Variablen, die mit "$" beginnen
€ Variables qui commencent par "$"

Mathematica stellt einige Variablen zur Verfügung. Diese beginnen mit dem Zeichen "$". Hier sind sie:
• *Mathematica* met quelques variables à disposition. Esses commencent par le signe "$". Les voici:

```
??$*
```

Einige Anwendungen:
• Quelques applications:

```
??$Line
```

```
$Line
```

```
$Version
```

```
$VersionNumber
```

```
$Letters
```

■ 5.12 Spezielle Formen
€ Formes spéciales

Mathematica-Zeichen sind Abkürzungen für Funktionen:
• Les signes de *Mathematica* sont des abréviations de fonctions:

```
??/
```

```
Alias["/"]
```

```
2/3 // N
```

```
Divide[2, 3] // N
```

```
Divide[2, 3]
```

```
Apply[Divide,{2,3}]
```

```
??Apply
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Uebungen € Exercices

5. Etwas Umgang mit *Mathematica*
€ Savoir un peu manier Mathematica

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 5 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 5.
WIR94/99

■ Aufgabe 1 € Problème 1

■ Ein Vorspann
€ Génériques

Sinus ohne ";" , damit Input-Lines erzeugt werden.
• Sinus sans ";" , pour créer des Input-Lines.

```
Plot[Sin[x],{x,0,6 Pi},
    AspectRatio -> 1/2,
    PlotLabel -> "Sin[x]",
    AxesLabel -> {"x", "y"},
    AxesOrigin -> {2 Pi, 1/2} ];
Plot[2x^3-7x^2-17x+10,{x,-6,6}];
Solve[2x^3-7x^2-17x+10==0,x]
```

■ Input-Lines auflisten
€ Faire une Liste des Input-Lines

Mit "??": • Avec "??":

```
??In
```

■ Aufgabe 2 € Problème 2

■ Auffinden von Kommandos
€ Trouver des commandements

Finde alle Kommandos mit den Worten "Matrix" und "Power" im Namen:
• Trouve tous les commandements qui ont les paroles "Matrix" et Power" dans leur nomes:

```
?*Matrix* *Power*
```

Oder: • Ou:

```
??*Matrix* *Power*
```

Oder: • Ou:

```
Print[Names["*Matrix*"]];
Names["*Power*"]
```

Kurze Neugier:
• Petite curiosité:

```
??Names
```

■ Aufgabe 3 € Problème 3

■ CPU-Zeit messen, 3 x 3-Matrix
€ Mesurer le temps CPU, matrice 3 x 3

Invertierung einer 3 x 3-Matrix. Postfix-Notation verwenden.
Erst Matrix generieren:
• Invertir une matrice * x 3. User la notation postfixe. Générer d'abort la matrice:

```
m = Table[Random[ ], {3}, {3}]; MatrixForm[m]
```

Dann Inverse rechnen und Zeit messen (Postfix-Notation):
• Ensuite calculer l'inverse et mesurer le temps (notation postfixe):

```
m // Inverse // MatrixForm // Timing
```

■ CPU-Zeit messen, 10 x 10-Matrix
€ Mesurer le temps CPU, matrice 10 x 10

Matrix generieren, Inverse rechnen und Zeit messen (Postfix-Notation):
• Générer une matrice, calculer l'inverse et mesurer le temps. (Notation postfixe):


```
Table[Random[ ], {10}, {10}] // Inverse; // Timing
```

■ Aufgabe 4 € Problème 4

■ (a) Fakultät berechnen:
€ Calculer des factoriaux

1 ! bis 5 ! • 1! à 5!

```
Do[Print[i!], {i, 5}]
```

■ (a) Grössere Fakultät berechnen:
€ Calculer des factoriaux plus grands:

1 ! bis 250 ! ***Achtung!*** Der Rechner kann damit unmöglich fertig werden.
Die Zahlen sind zu gross! Daher musst Du den Vorgang unterbrechen!
Versuche in **Action**, **Interrupt**, **Abort Calculation** oder **Interrupt Calculation**.
• 1 ! à 250 ! ***Attention!*** La calculatrice ne peut pas en venir à bout. Les nombres sont trop grands! Tu dois donc interrompre le processus! Essaie
dans **(Action) Kernel**, **Interrupt**, **Abort Calculation** ou **Interrupt Calculation**.

```
Do[Print[i!], {i, 42}]
```

Bitte nicht ausdrucken:
• S.v.p. ne pas imprimer:

```
(* Do[Print[i!], {i, 250}] *)
```

```
500!
```

■ Aufgabe 5 € Problème 5

■ Einige Variablen:
€ Quelques variables:

```
??$Line
```

```
??$Version
```

```
??$VersionNumber
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

6. Manipulation von Listen
€ Manipulation de listes

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 6 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 6.
WIR94/99

■ 6.0. Einleitung
€ Introduction

Listen - was sind Listen - was ist das?
• Des listes - qu'est - ce?

```
??List
```

Eingabe einer Liste:
• Entrer une liste:

```
list0 = {5, 4.78, a, 8x + y, Sin[x], D,
Integer, x->E}
```

Oft gibt *Mathematica* auch Listen aus. Beispiel:
• *Mathematica* sort souvent des listes. Exemple:

```
Solve[{2x + 5y == 19, 3x + 7y == 27}]
```

Hier ist eine Liste mit einer Liste ausgegeben worden.
• Ici on a sorti une liste qui contient une liste.

Ebenso z.B. bei "Options":
• Egalement p.ex. avec "Options":


```
Options[NIntegrate]
```

Hier ist eine Liste ausgegeben worden.
• Ici on a sorti une liste.

■ 6.1. Erzeugung von Listen
€ Générer des listes

■ 6.1.1. Befehle
€ Ordres

Zur Erzeugung von Listen stehen verschiedene Befehle zur Verfügung:
• Pour créer des listes on a plusieurs ordres à dispostion:

```
??Range
```

```
??Table
```

```
??Array
```

■ 6.1.2. Erzeugung einer Liste mit "Range"
€ Créer une liste avec "Range"

Beispiele: • Exemples:

```
Range[5]
```

```
Range[7.896]
```

```
Range[6,12]
```

```
Range[3.4, 8.7]
```

```
Range[3.4, 8.7, 1.2]
```

■ 6.1.3. Erzeugung einer Liste mit "Table"
€ Générer des listes avec "Table"

Beispiele: • Exemples:

```
Table[hoi,{6}]
```

```
Table[i!,{i, 6}]
```

```
Table[3i,{i, 6, 14}]
```

```
Table[i,{i, 4, 40, 5}]
```

```
Table[{i, i^2},{i, 5}]
```

```
Table[a[i] + b[j], {i, 1, 3}, {j, 4, 6}]
```

Das ist eine Liste von Listen, d.h. eine Matrix.
Zur besseren Darstellung verwenden wir "MatrixForm":
• C'est une liste de listes, c'est-à-dire une matrice.
Pour plus de clarté nous utilisons "MatrixForm"

```
MatrixForm[%]
```

Ein Iterator kann auch von einem andern abhängen:
• Un iterateur peut aussi dépendre d'un autre:

```
Table[a[i] + b[j], {i, 1, 3}, {j, i, 6}]
```

```
MatrixForm[%]
```

■ 6.1.4. "Array"
€ "Array"

Mit "Array" kann man eine symbolische Matrix erzeugen:
• Avec "Array" on peut créer une matrice symbolique:

```
Clear[a]
```

```
Array[a,{2, 3}]
```

```
MatrixForm[%]
```

```
a = 17; Array[a,{2, 3}] // MatrixForm
```

■ 6.2. Umordnen von Listen
€ Ranger autrement des listes

■ 6.2.1. Befehle
€ Ordres

```
??Sort
```

```
??Reverse
```

```
??RotateLeft
```

```
??RotateRight
```

```
??Permutations
```

```
??Drop
```

```
??Take
```

```
??First
```



```
??Last
```

```
??Part
```

```
??Rest
```

```
??Select
```

■ 6.2.2. Anwendungen
€ Applications

Hier einige Beispiele:
• Voici quelques exemples:

```
listA = Table[Random[Integer, {0, 12}],{16}]
```

```
Sort[listA]
```

```
Union[listA]
```

"Sort" sortiert die Liste, "Union" sortiert sie auch, eliminiert aber Duplikate von Elementen.
• "Sort" trie la liste, "Union" la trie aussi, mais élimine les doubles des éléments.

```
Reverse[listA]
```

```
RotateLeft[%,3]
```

```
RotateRight[%,4]
```

```
Permutations[{a, b, c}]
```

```
Permutations[{a, b, b}]
```

■ 6.3. Erweiterung und Verkürzung von Listen
€ Allonger et raccourcir des listes

■ 6.3.1. Befehle
€ Ordres

Probiere aus: • Essaie:

```
?Append
```

```
?AppendTo
```

```
?Drop
```

```
?Insert
```

```
?Prepend
```

```
?PrependTo
```

```
?Rest

?Take

?First

?Last

?Part

?Select
```

■ 6.3.2. Beispiele
€ Exemples

Probiere aus: • Essaie:

```
Print[listA];
Rest[listA]
```

Was ist passiert?
• Que s'est-il passé?

```
Drop[listA, 10]

Drop[listA, -5]

Drop[listA, {2}]

Drop[listA, {1}]

Drop[listA, {3}]

Drop[listA, {1,8}]

Drop[listA, {4,12}]

Take[listA,4]

Take[listA,-6]

Take[listA,{4,6}]

Append[listA, 99]

Prepend[listA, 100]

Insert[listA, 999, 4]

listA
```

Bei all diesen Operationen ist "listA" unverändert geblieben. Nicht aber bei den folgenden Operationen:
• Pendant toutes ces opérations "listA" n'a pas changé. Cela n'est pas le cas pour les opérations suivantes:

```
Print[listA];
AppendTo[listA, 2000]
```



```
PrependTo[listA, 555]
```

■ 6.4. Zählung der Elemente einer Liste
€ Compter les éléments d'une liste

■ 6.4.1. Befehle:
€ Ordres:

Probiere aus: • Essaie:

```
?Length

?Dimensions
```

■ 6.4.2. Anwendungen
€ Applications

```
Length[listA]

Length[{a, 4, 2^8, t^2 + 2t - 6, matrix}]

Length[{{a, 4, 2^8, t^2 + 2t - 6, matrix},
       listA}]

Dimensions[{{a, 4, 2^8, t^2 + 2t - 6, matrix},
           listA}]

Dimensions[{{a, 4, 2^8, t^2 + 2t - 6, matrix},
           listA}, 0]

Dimensions[{{a, 4, 2^8, t^2 + 2t - 6, matrix},
           listA}, 1]

Dimensions[{{a, 4, 2^8, t^2 + 2t - 6, matrix},
           listA}, 2]

MatrixForm[{{a, 4, 2^8, t^2 + 2t - 6, matrix},
           listA}]

Dimensions[{{a, 4, 2^8, t^2 + 2t - 6},
           {1, 2, 3, 4}}, 2]

MatrixForm[{{a, 4, 2^8, t^2 + 2t - 6},
           {1, 2, 3, 4}}]
```

■ 6.5. Zusammenfügen von Listen, Beziehungen zwischen Listen
€ Assemblage de listes, relations entre listes

■ 6.5.1. Befehle
€ Ordres

Probiere aus: • Essaie

```
?Complement

?Union

?Join

?Intersection
```

■ 6.5.2. Anwendungen
€ Applications

```
Complement[{1, 2, 3, 4, 5},{1, 2, 3}]

Intersection[{a, b, c, d, e, f},
               {d, e, f, g, h, i, j, k}]

Union[{a, b, c, d, e, f},
                     {d, e, f, g, h, i, j, k}]

Union[{1, 1, 1, 1, 2, 1, 3, 3, 3, 2, 4, 2, 4, 4,
               3, 3, 3}]

Join[{a, b, c, d, e, f},{d, e, f, g, h, i, j, k}]

Join[{1, 2, 1, 1, 2, 2},{1, 2, 2, 1, 3, 2, 1,
               3, 3}]

Union[{1, 2, 1, 1, 2, 2},{1, 2, 2, 1, 3, 2, 1,
               3, 3}]
```

■ 6.6. Veränderung der Form einer Liste
€ Changer la forme d'une liste

■ 6.6.1. Befehle
€ Ordres

Probiere aus: • Essaie:

```
?Flatten
```



```
?Partition

?Transpose
```

■ 6.6.2. Anwendungen
€ Applications

```
vector1 = Range[10]

matrix1 = Partition[vector1, 5]

MatrixForm[matrix1]

matrix2 = Partition[vector1, 4]

matrix3 = Partition[vector1, 3]

matrix4 = Partition[vector1, 2.5]
```

Was ist passiert?
• Que s'est-il passé?

```
Partition[{1, 35, 432, 2, 5467, 4, 3, 987, 87},
                                                3]

Partition[{1, 2, 3, 4, 5, 6, 7}, 3, 1]

Partition[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9},
           {a, b, c}}, 2, 1]

Print[matrix1];
Flatten[matrix1]

m = {{1, 2, 3, a},{4, 5, 6, b}}

Transpose[m]

Print[MatrixForm[m]];
MatrixForm[Transpose[m]]
```

■ 6.7. Elemente herauspicken
€ Choisir quelques éléments

■ 6.7.1. Befehle
€ Ordres

Oben sind schon "First", "Last" und "Part erwähnt worden.
• Plus haut on a déjà nommé "First", "Last", et "Post".

■ 6.7.2. Anwendungen
€ Applications

```
liste1 = {a, b, c, d, e, f, g}

First[liste1]

Last[liste1]

liste1[[4]]

Print[{liste1[[2]],liste1[[3]],liste1[[4]],
       liste1[[5]]}]

liste1[[ Range[2, 5] ]]

array1 = Array[a, {2, 2}]

array1[[2]]

array1[[1, 2]]
```

■ 6.8. Auswahl von Daten
€ Choisir des données

■ 6.8.1. Befehle
€ Ordres

Probiere aus: • Essaie:

```
?Select

?*Q

?DigitQ

?IntegerQ

?LetterQ

?LowerCaseQ

?MachineNumberQ

?MatrixQ

?NameQ

?NumberQ

?OddQ

?EvenQ
```



```
?PrimeQ

?UpperCaseQ

?ValueQ

?VectorQ

?Positiv*

?Negat*
```

■ 6.8.2. Anwendungen
€ Applications

Probiere aus: • Essaie:

```
MemberQ[liste1, 3]

MemberQ[liste1, a]

menge = {-4, 3, 5.678, emil, Emma, 4, 0, -2, -4,
          Pi, 2^(1/2)}

Select[menge, Positive]

Select[{a, b, c, d, x, y, z}, Positive]
```

Bei "Select" steht also hinten das Selektionskriterium. Solche Kriterien kann man selbst definieren. Beispiel:
• Quant à "Select", le critère de sélection se trouve à la fin. On peut définir soi-même de tels critères. Exemple:

```
?Precision

?===

exakt[x_] := Precision[x] === Infinity

Select[{-6, 0, 3, 3.1414333, 1.111111, Pi, E,
       N[Sqrt[2], 10], 5.66}, exakt]
```

■ 6.9. Rechnen mit Listen
€ Calculer avec des listes

■ 6.9.1. Befehle
€ Ordres

```
?Sum

?Plus

?Apply
```

■ 6.9.2. Wie man's macht:
€ C'est ainsi que cela se fait:

```
Sum[n^2, {n, 7}]
```

```
meineDaten = {3.4, 6.1, 5.3}
```

```
Sum[meineDaten[[n]], {n, Length[meineDaten]}]
```

```
Plus[1, 1, 1, 1, 1]
```

```
Plus[1, 2, 3, 4, 5]
```

```
Plus[a, b, c, d]
```

```
Plus[{a, b, c, d}]
```

```
??Plus
```

"Plus" hat das Attribut "Listable".
• "Plus" a l'attribut "Listable".

```
??Listable
```

"Plus" auf eine Liste angewandt addiert nur ein Element, nämlich die Liste selbst. Wie nun "Plus" aber auf die Elemente anwenden? - So:
• "Plus" appliqué à une liste additionne seulement un élément, c'est-à-dire la liste même. Comment appliquer "Plus" aux éléments? - Ainsi:

```
Apply[Plus, {a, b, c, d}]
```

Was geht nun schneller, "Sum" oder "Plus" ?
• Qu'est-ce qui est plus rapide, "Sum" ou "Plus"?

```
Timing[Apply[Plus, Range[500]]]
```

```
Timing[Sum[n, {n, 500}]]
```

■ 6.10. Auf Listen anwendbare arithmetische Funktionen
€ Fonctions arithmétiques applicables à des listes

■ Beispiele
€ Exemples

Wir wenden "Log" auf eine Liste an:
• Nous appliquons "Log" à une liste:

```
??Log
```



```
Log[{1, 2, 3, 4, 5, 6, 7}]
```

```
N[%]
```

Oder mit "Sinus":
• Ou avec "Sinus":

```
Sin[{1, 2, 3, 4, 5, 6, 7}]
```

```
Sin[{1, 2, 3, 4, 5, 6, 7}] // N
```

Oder "Plus":
• Ou "Plus":

```
?Plus
```

```
?*Form*
```

```
FullForm[a + b]
```

```
Plus[{1, 2, 3, 4}, {a, b, c, d}]
```

```
{1, 2, 3, 4} + {a, b, c, d}
```

Attribute einiger Funktionen:
• Attributes de quelques fonctions:

```
Attributes[{Exp, Log, Sin, Cos, Plus, Times}]
```

```
Attributes[Attributes]
```

"Attributes" hat also auch das Attribut "Listable".
• "Attributes" a donc aussi l'attribut "Listable".

■ 6.11. Listable und Map
€ Listable et Map

■ 6.11.1. Befehle
€ Ordres

Probiere: • Essaie:

```
??Map
```

```
?Listable
```

```
?Apply
```

■ 6.11.2. Anwendungen
€ Applications

Es gibt zwei Wege, eine Funktion auf eine Liste anzuwenden: mit "Listable" als Attribut und mit der Funktion "Map".
Probiere:
• Il y a deux façons d'appliquer une fonction à une liste: avec l'attribut "List" et avec la fonction "Map". Essaie:

```
f[{1, 2, 3, 4, 5, 6}]
```

```
?f
```

f ist nicht "Listable".
• F n'est pas "Listable"

```
Map[f, {1, 2, 3, 4, 5, 6}]
```

So geht's! Oder so:
• Voilà! Ou ainsi:

```
Map[f, a x^2 + b x + c]
```

Map wirkt auf die Teile eines Ausdrucks! Beachte:
• Map a un effet sur les parties d'une expression! Remarque:

```
Map[f, {a, b, c, d, e}]
```

```
Apply[f, {a, b, c, d, e}]
```

Mit Apply wird f angewandt auf die Elemente der Menge als Ganzes, während mit Map diejenige Menge gebildet wird, die als neue Elemente die Funktionswerte der einzelnen alten Elemente enthält.
• Avec Apply on applique f aux éléments de l'ensemble comme un tout, tandis que avec Map on forme l'ensemble qui contient en tant qu'éléments nouveaux les valeurs des fonctions des vieux éléments.

Statt Apply und Map kann man auch /@ und @@ benutzen. Beispiele:
• On peut employer aussi /@ et @@ au lieu de Apply et Map. Exemple:

```
f/@{a, b, c, d, e}
```

```
f@@{a, b, c, d, e}
```

```
Sin/@{0, Pi, 2 Pi, 2, 3, 4, 5, 6}
```

```
f1[x_, y_,z_]:= Sin[x z^2] Cos[y]
```

```
f1@@{1, 2, 3}
```


■ 6.12. Formatierung von Listen
€ Formater des listes

■ 6.12.1. Befehle
€ Ordres

Probiere: • Essaie:

```
?ColumnForm
```

```
?MatrixForm
```

```
?TableForm
```

■ 6.12.2. Beispiele
€ Exemples

```
?Binomial
```

```
pascal[n_]:= Table[Binomial[n, i],{i, 0, n}]
```

```
pascal[5]
```

```
Table[pascal[n],{n, 0, 5}]
```

```
ColumnForm[Table[pascal[n],{n, 0, 5}]]
```

```
ColumnForm[Table[pascal[n],{n, 0, 5}], Center]
```

```
ColumnForm[Table[pascal[n],{n, 0, 5}], Right]
```

```
ColumnForm[Table[pascal[n],{n, 0, 5}], Left]
```

```
TableForm[Table[pascal[n],{n, 0, 5}]]
```

```
MatrixForm[Table[pascal[n],{n, 0, 5}]]
```

```
MatrixForm[Table[n - i,{i,5},{n, 5}]]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

6. Manipulation von Listen
€ Manipulation de listes

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 6 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 6.
WIR94/99

■ Aufgabe 1 € Problème 1

■ **Erzeugung einer Liste der ersten 29 ungeraden natürlichen Zahlen**
€ Générer une liste des premiers 29 nombres naturels impairs

Mit Range • Avec Range

```
Range[1,20,2]
```

Mit Table • Avec Table

```
oddList = Table[2i-1,{i,10}]
```

■ Aufgabe 2 € Problème 2

■ **Herausnehmen und weglassen von 4 Elementen**
€ Enlever 4 éléments

Die ersten 4 Elemente nehmen:
• Prendre les 4 premiers éléments:

```
Take[oddList,4]
```

Die letzten 4 Elemente weglassen:
• Supprimer les 4 derniers éléments:

```
Drop[oddList, -4]
```


■ Aufgabe 3 € Problème 3

■ **Funktionen "Dimension" und "Length",**
Interpretation des Resultats
€ Fonctions "Dimension" et "Length",
interprétation des résultats

Eingabe der Matrizen
• Entrer les matrices

```
m = {{2, 4, 6}, {5, 7, 9}}; n = {{1, 2, 3}, {4,
                                              5}};
Print[MatrixForm[m]]; Print["***"]; Print[MatrixForm[n]];
```

Dimension • Dimension

```
Dimensions[m]
```

```
Dimensions[n]
```

Länge • Longueur

```
Length[m]
```

```
Length[n]
```

■ Aufgabe 4 € Problème 4

■ **Einige Listenmanipulationen**
€ Quelques manipulations de listes

■ **(a) Listen generieren**
€ Générer des listes

Zwei Listen kreieren:
• Créer deux listes

```
li1 = Table[Random[Integer,{0,10}],{i,10}]
```

```
li2 = Table[Random[Integer,{0,10}],{i,10}]
```

Als Mengen:
• Comme ensembles:

```
Union[li1]
```

```
Union[li2]
```

■ (b) Sortieren
€ Trier

Sortieren von li1:
• Trier li1:

```
Sort[li1]
```

■ (c) Anfügen
€ Ajouter

An li1: • A li1:

```
Append[li1, 5]
```

■ (d) Schnittmenge
€ Intersection

von li1 und li2:
• de li1 et li2:

```
Intersection[li1, li2]
```

■ Vereinigung
€ Réunir

von li1 und li2:
• li1 et li2:

```
Union[li1, li2]
```

■ (e) Zusammenhängen
€ Relier

von li1 und li2:
• li1 et li2:

```
Join[li1, li2]
```


■ (f) Verschiedene Elemente
€ Différents éléments

in li1 und li2:
• dans li1 et li2:

```
Complement[Union[li1, li2],Intersection[li1, li2]]
```

■ (f) Differenz
€ Différence

li1 ohne li2:
• li1 sans li2:

```
Complement[li1, li2]
```

■ (f) Komplement
€ Complément

von li1 und li2 bezüglich Range[10]:
• de li1 et li2 par rapport à Range[10]:

```
Complement[Range[10],Union[li1, li2]]
```

■ Aufgabe 5 € Problème 5

■ Test, ob alle Elemente einer Liste dieselben sind
€ Test, si tous les öléments d'une liste sont les mêmes

Liste von Listen von Zahlen:
• Liste de listes de nombres:

```
myList = {{-1,-1,2},{-1,-1,2},{-1,-1,2},{-1,-1,2}}
```

```
Length[Union[myList]]
```

Liste von Polynomen:
• Liste de polynômes:

```
myList = {x^2-1,x^2-1,x^2-1,x^2-1,(x-1)(x+1)}
```

```
Length[Union[myList]]
```

■ Aufgabe 6 € Problème 6

■ Listenmanipulationen
€ Manipulations de listes

■ (a) Liste generieren
€ Générer une liste

Liste generieren mit 16 Zahlen zwischen 0 und 100:
• Générer une liste de 16 nombres entre 0 et 100:

```
li3 = Table[Random[Integer,{0,100}],{i,16}]
```

■ (b) Partitionen
€ Partitions

mit 4 Teilmengen
• avec 4 sous-ensembles

```
li4 = Partition[li3,4]
```

Matrizenform:
• Forme de matrice:

```
MatrixForm[li4]
```

■ (c) 3. Zeile der Matrix
€ 3. ligne de la matrice

```
li4[[3]]
```

■ (d) 3. Spalte der Matrix
€ 3. colonne de la matrice

```
Transpose[li4][[3]]
```

```
Transpose[li4][[3]] // MatrixForm
```

■ (e) Mittelwert und Median
€ Valeur moyenne et médiane

Mittelwet: • Valeur moyenne:


```
mean = Sum[li3[[i]],{i,1,Length[li3]}]/Length[li3]
```

Oder: • Ou:

```
Apply[Plus,li3]/Length[li3]
```

Median: • Médiane:

Erst Liste sortieren:
• D'abort trier la liste:

```
sLi3 = Sort[li3]
```

Mittleres Element resp. Durchschnitt der beiden mittleren Elemente rechnen,
erst Funktion definieren:
• Calculer l'élément du milieu, resp. la moyenne des deux éléments, d'abort définir la fonction:

```
median[x_List] := (x[[Ceiling[ Length[x]   /2]]] +
                   x[[  Floor[(Length[x]+2)/2]]]
          ) / 2
```

Versuche, dieses Programm zu begreifen!
• Essaie de comprendre ce programme!

Rechnen: • Calculer la médiane:

```
median[sLi3]
```

Liste mit ungerader Anzahl Elemente definieren, sortieren:
• Définir une liste avec nombre impair d'éléments, trier:

```
sLi4 = Sort[Append[li3,99]]
```

Median rechnen:
• Calculer la médiane:

```
median[sLi4]
```

■ Aufgabe 7 € Problème 7

■ Mit "Select" Elemente herauspicken:
€ Choisir avec "Select" des éléments:

Liste: • Liste:

```
li3
```

Programm: • Programme:

```
greater50[x_]:= x > 50
```

Anwenden: • Appliquer:

```
Select[li3, greater50]
```

■ Aufgabe 8 € Problème 8

■ **Spiel mit "Apply" und "Map":**
€ Jeu avec "Apply" et "Map":

Liste eingeben:
• Entrer une liste:

```
Clear[myList]; myList = {{a,b},{c,d}}
```

(a) ausprobieren:
• essayer:

```
Apply[f, myList]
```

(b) ausprobieren:
• essayer:

```
Map[f, myList]
```

(c) On-line help:
• On-line help:

```
??MapAt
```

```
??MapAll
```

(d) ausprobieren:
• essayer:

```
MapAt[f, myList, {2}]
```

(e) ausprobieren:
• essayer:

```
MapAll[f, myList]
```

■ Aufgabe 9 € Problème 9

■ **Schreibe ein *Mathematica*-Programm, das aus zwei Zahlenreihen positionsweise immer die kleinere heraussucht**
€ Ecris un programme de *Mathematica* qui, de deux ensembles de nombres, choisit pour chaque position la plus petite

Zahlenreihen: • Ensembles de nombres:


```
r1 = {15, 17, 18, 32, 29}
```

```
r2 = {14, 18, 22, 29, 26}
```

Programm: • Programme:

```
Map[Min, Transpose[{r1, r2}]]
```

Dazu Details:
• Quelques détails:

```
Transpose[{r1, r2}] // MatrixForm
```

■ Aufgabe 10 € Problème 10

■ **Eine Aufgabe mit 3D-Graphik**
€ Un problème avec des graphiques 3D

Generiere eine Liste von 80 Punkten im 3-dimensionalen Raum:
• Générer une Liste de 80 points dans l'espace à trois dimensions

```
li5 = Table[Table[Random[Integer,{0,100}],{i,3}],
       {j,80}]
```

```
MatrixForm[%]
```

Graphik-Objekte herstellen:
• Faire des objets graphiques

```
punkte = Map[Point, li5]
```

Ploten: • "Ploter":

```
Show[Graphics3D[{PointSize[0.025], punkte}],
                        Axes -> Automatic];
```

```
Show[Graphics3D[{RGBColor[0.5,0.5,0.5],
            PointSize[0.025], punkte}],
                        Axes -> Automatic];
```

■ Aufgabe 11 € Problème 11

■ **Probiere in Aufgabe 10 die Graustufe vom Punkt abhängig zu machen, so dass jeder Punkt eine andere Graustufe bekommt!**
€ Prends le problème 10 et essaie de rendre l'intensité du gris dépendante du point, ainsi que chaque point aie une autre intensité de gris

Eine mögliche Lösung:
• Une solution possible:

```
Show[Graphics3D[
                Table[{
                    RGBColor[i/100,(i/100)^2,(i/100)^3],
                    PointSize[0.025+i/1000],
                    punkte[[i]]
                      },{i,Length[punkte]}]
                ],
                        Axes -> Automatic
    ];
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Kurs € Cours

7. Probleme zum Thema "Zuordnungen und Regeln"
€ Problèmes au sujet des "applications et règles"

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 7 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 7.
WIR94/99

Es geht hier darum zu zeigen, wie ein Name einem Wert oder einer Operation zugewiesen wird. Oder wie Symbole in Ausdrücken durch Werte ersetzt werden.
• Il s'agit de montrer ici comme un nom est assigné à une valeur ou à une opération. Ou comme on remplace des symboles dans des expressions par des valeurs.

■ 7.1. Zuordnungen
€ Applications

■ 7.1.1. Allgemeines
€ Généralités

Namen bestehen aus Buchstaben, Ziffern oder "$".
Eingebaute Mathematica-Namen beginnen mit Grossbuchstaben oder "$".
Namen mit "$" am Anfang sind globale Variablen. Probiere aus:
• Les noms sont formés de lettres, chiffres ou "$". Les noms incorporés de *Mathematica* commencent par une majuscule ou par "$". Les noms qui commencent par "$" sont des variables globales. Essaie:

```
a = 5
```

```
a + 4
```

Statt Werte kann man auch Ausdröcke zuordnen und damit rechnen:
• Au lieu des valeurs on peut aussi assigner des expressions et calculer avec celles-ci:

```
a = 3x + y
```

```
1/a
```

"." heisst "missing value". Weist man "." zu, so ist die Variable "leer":
• "." veut dire "missing value". Si on assigne ".", la variable est "vide":

```
a = .
```

```
a
```

■ 7.1.2. Unmittelbare und verzögerte Zuordnungen
€ Applications immédiates et retardées

"=" oder "Set" bewirkt eine sofortige Zuordnung. Probiere aus:
• "=" ou "Set" produisent une application immédiate. Essaie:

```
?Set
```

```
?=
```

```
Set[a, 33444]
```

```
a
```

```
random1 = Random[]
```

```
2
```

```
3 + random1
```

":=" oder "SetDelayed" bewirkt die Zuordnung erst dann, wenn die nachstehende Funktion aufgerufen wird. Probiere:
• ":=" ou "SetDelayed" produisent l'application seulement, quand la fonction suivante est appelé. Essaie:

```
?:=
```

```
?SetDelayed
```

```
random2 := Random[]
```

Kein Output, da random2 nicht gerufen, sondern erst gesetzt worden ist.
Vergleiche und erkläre:
• Pas de "Output" car random2 n'a pas été appelé, mais seulement été placé. Compare et explique:

```
Table[random1,{5}]
```

```
Table[random2,{5}]
```

Was ist der Unterschied und wieso tritt er auf? (Du kannst auch mehrmals probieren.) Anwendung auf die Definition von Funktionen:
• Quelle est la différence et apparaît-il? (Tu peux essaier plusieurs fois.) Application à la définition de fonctions:

```
Remove[f];Remove[g];
```

a) Unmittelbare Zuweisung. Beobachte, was b bewirkt:
• Application immédiate. Observe ce que produit b:

```
b = 3;
f[x_] = x + b
```


Veränderung des Wertes und Aufruf: Was bewirkt b?
• Changement des valeurs et appel: Que produit b?

```
b = 100;
f[5]
```

b) Verzögerte Zuweisung. Beobachte, was b bewirkt:
• Application retardée. Observe ce que produit b:

```
b = 3;
g[x_] := x + b
```

Veränderung des Wertes und Aufruf: Was bewirkt b?
• Changement de la valeur et de l'appel: Que produit b?

```
b = 100;
g[5]
```

Abruf der Definitionen:
• Appel des définitions:

```
?f
```

```
?g
```

■ 7.1.3. Mehrfache Zuordnungen
€ Applications multiples

In einem Schritt können mehrere Zuordnungen gemacht werden. *Mathematica* arbeitet die Zuordnungen von rechts nach links ab:
• On peut faire plusieurs applications en une démarche. *Mathematica* achève les applications de droite à gauche:

```
a1 = a2 = a3
```

Oder listenweise:
• Ou par listes:

```
{u, v} = {4, 15}
```

```
{s, t} = {u, v}
```

```
{t, s}
```

■ 7.1.4. Einschub für 7.4.1.
(dort nicht anwendbar, wegen zu grossem Output)
€ Insertion pour 7.4.1.
(ne peut pas être appliqué à cause du trop grand output)

```
??Out
```

■ 7.1.5. Rekursive Zuordnungen
€ Application récursive

Durch rekursive Zuweisung (rekursive Funktion) kann eine Liste erzeugt werden. Hier das Beispiel der Fakultäten:
• Par une application (fonction) récursive on peut créer une liste. Voici l'exemple pour des factoriaux:

```
Remove[fac];
fac[1]  = 1;
fac[n_]:= fac[n] = n fac[n-1];

?fac

Table[fac[n],{n,1,5}]

fac[50];
Timing[fac[50]]
```

Probiere: • Essaie:

```
Remove[fak];
fak[1]  = 1;
fak[n_]:= n fak[n-1];
Table[fak[n],{n,1,5}]

fak[50];
Timing[fak[50]]
```

Wieso braucht *Mathematica* das zweite Mal weniger Zeit? Das erste Mal speichert es wegen der Zwischenzuweisung alle Werte ab und kann den gewünschten Wert dann aus dem Speicher holen. Das zweite Mal muss das Programm rechnen!
• Pourquoi *Mathematica* emploie-t-il moins de temps la deuxième fois? La première fois il mémorise toutes les valeurs à cause des applications intermédiaires et peut sortir la valeur desirée de la mémoire. La deuxième fois le programme doit calculer!

■ 7.2. Löschen der Wertzuweisung
€ Effacer l'assignation des valeurs

■ 7.2.1. Missing value einsetzen:
€ Appliquer missing value

Beispiel: • Exemple:

```
Remove[x] ; x = 1 ; Print["1)   ", x] ;
x = . ; Print["2)   ",x]

??x

x = 1 ; Print["1)   ", x]

??x

x = . ; Print["2)   ",x]
```



```
??x
```

■ 7.2.2. Funktion löschen:
€ Effacer la fonction

```
?fak

fak[1] = .

fak[n_] = .

?fak
```

Zwar ist kein Wert mehr zugewiesen, doch die Funktion ist noch bekannt.
• Aucune valeur n'est plus assignée, cependant la fonction est encore connue.

Anderes Beispiel:
• Autre exemple

```
kubus[x_] = x^3

?kubus

Clear[kubus]

?kubus
```

Zwar ist keine Funktionsvorschrift mehr zugewiesen, doch der Name der Funktion ist noch bekannt.
• Aucune prescription de fonction n'est assignée. Cependant le nom de la fonction est encore connue.

```
Attributes[kubus] = {Listable}

Clear[kubus]

?kubus
```

Sogar das Attribut ist noch bekannt.
• Même l'attribut est encore connue.

```
ClearAll[kubus]

?kubus
```

Jetzt ist das Attribut nicht mehr bekannt.
• Maintenant l'attribut n'est plus connue.

```
Remove[kubus]

?kubus
```

Jetzt ist auch der Name nicht mehr bekannt.
• Maintenant même le nom n'est plus connue.

```
Remove[Apply]
```

(Hier handelt es sich um einen *Mathematica*-Namen!)
• (Ici il s'agit d'un nom de *Mathematica*))

■ 7.2.3. Probleme mit Packages:
€ Problèmes de Packages:

Es kann passieren, dass man eine Funktion aus einem Package aufruft, bevor das Package geladen worden ist. Nach dem Aufruf erscheint vielleicht eine Meldung, worauf man dann den Aufruf des Packages nachholt. Da aber vom ersten Aufruf her die Funktion schon registriert ist, lässt sie sich jetzt nicht einfach überschreiben. Man muss sie erst wieder löschen. Die mit dem Package geladene Funktion bleibt dann bestehen. Beispiel:
• Il peut arriver qu'on appelle une fonction d'un package avant le changement du package. Après l'appel il apparaît peut-être un message de rattraper ˚'appel du package. Mais comme la fonction a déjà été enregistrée au premiers appel, elle ne se laisse pas surcharger maintenant. Il faut d'abort l'effacer. La fonction chargée du Package reste. Exemple:

```
Show[Graphics[{CMYColor[0.6,0.5,0.4],
        Point[{0,0}]}]];

Needs["Graphics`Colors`"]

?CMYColor

Remove[CMYColor]

?CMYColor
```

■ 7.2.4. Löschen aller eigenen Variablen und Funktionen:
€ Effacer toutes les propres variables et fonctions:

Folgender Befehl löscht alle globalen Variablen und Funktionen, die mit einem Kleinbuchstabe oder mit "$" beginnen und nicht"protected" sind.
• L'ordre suivant efface toutes les variables et fonctions globales qui commencent par une minuscule ou par "$" et qui ne sont pas "prodected".

```
?fak

Remove["Global`@*"]

?fak
```

Der folgende Befehl löscht alle Variablen und Funktionen, die nicht"protected" sind. Achtung vor der Anwendung!
• L'ordre suivant efface toutes les variables et fonctions qui ne sont pas "prodected".

```
(*   Remove["@*"]   *)
```

Falls was schief gegangen ist, kannst Du im Menu unter ("Action") "Kernel", "Kernels and Tasks..." einen neuen Kernel starten.
• Si quelque chose n'a pas fonctionné, tu peux démarrer un nouveau Kernel dans le menu sous ("Action") "Kernel", "Kernels and Tasks...".

■ 7.3. Regeln
€ Règles

■ 7.3.1. Allgemeines
€ Généralités

```
?/.

?ReplaceAll

?Replace
```

In *Mathematica* gibt es also die Möglichkeit, nach der Berechnung durch Anwendung einer Ersetzungsregel einen Ausdruck durch etwas anderes zu ersetzen. Beispiele:
• Dans *Mathematica* il y a la possibilité, après un calcul où on emploie une règle de remplacement, de remplacer une expression par autre chose. Exemple:

```
x + 7y

x + 7y /. x->4

x + 7y /. {x->4, y->3}

x + 7y
```

Die Ersetzung ist also nicht bleibend! Beachte:
• Le remplacement n'est pas définitif! Observe:

```
3x + 7x /. {3x->y, y->3}
```

Da zuerst gerechnet wird, kommt die Ersetzungsregel gar nie zur Ausführung.
• Comme on commence par le calcul, la règle de remplacement n'arrive pas à être utilisée.

■ 7.3.2. Verzögerte Ausführung der Regel
€ Exécution retardée de la règle

Vergleiche: • Compare:

```
?->

?:>
```

Man hat also eine ähnliche Situation wie bei "=" und ":=". Beispiele:
• On a donc une situation semblable comme pour "=" et ":=". Exemple:

```
Table[x, {6}]

Table[x, {6}] /. x-> Random[]
```

Offenbar ist erst gerechnet worden. Dann wurde Random[ ] einmal ausgeführt, darauf zugewiesen. x bleibt leer:
• Apparemment on a d'abord calculé. Ensuite on exécute une fois Random[ ], puis assigné. x reste vide.

```
x
```

```
Table[x, {6}] /. x:> Random[]
```

Hier ist demnach Random[ ] jeweils vor der nächsten Zuweisung verzögert ausgeführt worden.
• On a donc retardé chaque fois l'exécution de Random[ ] avant la prochaine assignation.

■ 7.3.3. Wiederholte Ersetzung € Remplacement répété:

Studiere: • Etudie:

```
?//.
```

Vergleiche: • Compare:

```
log[a b c d] /. log[x_ y_] -> log[x] + log[y]
```

```
log[a b c d] //. log[x_ y_] -> log[x] + log[y]
```

■ 7.3.4. Ersetzung bei graphischen Optionen € Remplacement lors d'options graphiques

Studiere, welche Optionen im nächsten Beispiel verzögert und welche gewöhnlich sind:
• Etudie quelles options, dans l'exemple suivant, sont retardées et lesquelles sont normales:

```
Options[ContourPlot]
```

■ 7.3.5. Eine Anwendung: Labels bei Plots € Une application: "Labels" des "Plots"

Aufgabe: Jeder Plot soll als Label das Kommando erhalten, das ihn generiert hat.
Studiere, was nun folgt:
• Problème: Chaque Plot reçoit en tant que Label le commandement qui l'a généré. Etudie ce qui en suit:

```
?$Line
```

```
$Line
```

```
?InString
```

```
?ToString
```

```
?StringJoin
```

```
?SetOptions
```



```
SetOptions[Plot,PlotLabel :> StringJoin[
           "In[", ToString[$Line], "] := ",
           InString[$Line]
           ]];
Plot[Sin[x],{x, 0, 2 Pi}];
```

```
Plot[Cos[x],{x, 0, 2 Pi}];
```

■ 7.3.6. Vereinfachung von trigonometrischen Ausdrücken € Simplification d'expressions trigonométriques

Dazu gab es ein Package. (Neu bei Build-in Functions)
• Pour cela il existait un Package. (Nouveau avec Build-in Functions)

```
Needs["Algebra`Trigonometry`"] (* old ! *)
```

```
??TrigReduce
```

```
??TrigReduceRel
```

Offenbar kann (konnte) man Dinge nicht ansehen, die mit "Private" spezifiziert sind (waren). Beachte die Anwendung:
• Apparemment on ne peut (pouvait) pas regarder les choses qui sont (étaient) spécifiées par "Private". Tiens compte de l'application:

```
TrigReduce[Sin[x + Pi]]
```

Beachte die Ersetzungsregeln im Aufbau des folgenden Programms.
(Die Sache wird in einem späteren Kapitel erklärt. Hier ist nur Text - so läuft es nicht.)
• Considère les règles de remplacement dans la construction (structure) du programme suivant. (La chosesera expliquée dans un chapitre ultérieurs. Ici ol n'y a que du teste - cela ne va (tourne) pas ainsi.)

```
BeginPackage["Algebra`Trigonometry`", "Global`"]

TrigCanonical::usage = "TrigCanonical[expr] is obsolete.
      Its functionality is now built-in."
TrigExpand::usage = "TrigExpand[expr] is obsolete.
      Its functionality is provided by Expand[expr, Trig->True]."
TrigFactor::usage = "TrigFactor[expr] tries to write sums of trigonometric
      functions as products."
TrigReduce::usage = "TrigReduce[expr] writes trigonometric functions of
      multiple angles as sums of products of trigonometric functions of
      that angle."
TrigReduce::notes = "TrigReduce simplifies the arguments of trigonometric
      functions. It is, in a way, the inverse of TrigExpand."
TrigToComplex::usage = "TrigToComplex[expr] writes trigonometric functions
      in terms of complex exponentials."
ComplexToTrig::usage = "ComplexToTrig[expr] writes complex exponentials
      as trigonometric functions of a real angle."

Begin["`Private`"]
{`x, `y, `r, `n, `m, `a, `b, `c, `i, `e};
```

```
TrigCanonical[e_] := e

TrigExpand[___] := $Failed /;
   Message[TrigExpand::obsfn, TrigExpand, Expand]

TrigExpand[e_]:=Expand[e,Trig->True]

`TrigFactorRel = {
   a_. Sin[x_] + a_. Sin[y_] :> 2 a Sin[x/2+y/2] Cos[x/2-y/2],
   a_. Sin[x_] + b_. Sin[y_] :> 2 a Sin[x/2-y/2] Cos[x/2+y/2] /; a+b == 0,
   a_. Cos[x_] + a_. Cos[y_] :> 2 a Cos[x/2+y/2] Cos[x/2-y/2],
   a_. Cos[x_] + b_. Cos[y_] :> 2 a Sin[x/2+y/2] Sin[y/2-x/2] /; a+b == 0,
   a_. Tan[x_] + a_. Tan[y_] :> a Sin[x+y]/(Cos[x] Cos[y]),
   a_. Tan[x_] + b_. Tan[y_] :> a Sin[x-y]/(Cos[x] Cos[y]) /; a+b == 0,

   a_. Sin[x_] Cos[y_] + a_. Sin[y_] Cos[x_] :> a Sin[x + y],
   a_. Sin[x_] Cos[y_] + b_. Sin[y_] Cos[x_] :> a Sin[x - y] /; a+b == 0,
   a_. Cos[x_] Cos[y_] + b_. Sin[x_] Sin[y_] :> a Cos[x + y] /; a+b == 0,
   a_. Cos[x_] Cos[y_] + a_. Sin[x_] Sin[y_] :> a Cos[x - y]

}
TrigFactorRel = Dispatch[TrigFactorRel]
Protect[TrigFactorRel]

TrigFactor[e_] := FixedPoint[(# /. TrigFactorRel)&, e]

`TrigReduceRel = {

   Cos[n_Integer x_] :> 2^(n-1) Cos[x]^n +
         Sum[ Binomial[n-i-1, i-1] (-1)^i n/i 2^(n-2i-1) Cos[x]^(n-2i),
           {i, 1, n/2} ]              /; n > 0,

   Sin[m_Integer?OddQ x_] :>
         Block[{`p = -(m^2-1)/6, `s = Sin[x], `k},
            Do[s += p Sin[x]^k;
            p *= -(m^2 - k^2)/(k+2)/(k+1),
            {k, 3, m, 2}];
            m s]                                  /; m > 0,

   Sin[n_Integer?EvenQ x_] :>
         Sum[ Binomial[n, i] (-1)^((i-1)/2) Sin[x]^i Cos[x]^(n-i),
           {i, 1, n, 2} ]                  /; n > 0,

   Tan[n_Integer x_] :> Sin[n x]/Cos[n x],

   Sin[x_ + y_] :> Sin[x] Cos[y] + Sin[y] Cos[x],
   Cos[x_ + y_] :> Cos[x] Cos[y] - Sin[x] Sin[y],
   Tan[x_ + y_] :> (Tan[x] + Tan[y])/(1 - Tan[x] Tan[y]),
```



```
   Sin[r_Rational x_] :> (Sin[Numerator[r] `symb] /. TrigReduceRel /.
         `symb -> x/Denominator[r])          /; Numerator[r] != 1,
   Cos[r_Rational x_] :> (Cos[Numerator[r] `symb] /. TrigReduceRel /.
         `symb -> x/Denominator[r])          /; Numerator[r] != 1,

   Tan[x_/2] :> (1 - Cos[x])/Sin[x],
   Cos[x_/2]^(n_Integer?EvenQ) :>
                  ((1 + Cos[x])/2)^(n/2),
   Sin[x_/2]^(n_Integer?EvenQ) :>
                  ((1 - Cos[x])/2)^(n/2),
   Sin[x_/2]^n_. Cos[x_/2]^m_. :> Tan[x/2]^n              /; m == -n,
   Sin[r_ x_.] Cos[r_ x_.]      :> Sin[2 r x]/2  /; IntegerQ[2r]

}
TrigReduceRel = Dispatch[TrigReduceRel]

TrigReduce[e_] := e //. TrigReduceRel

`TrigToComplexRel = {
   Sin[x_] :> -I/2*(-E^(-I*x) + E^(I*x)),
   Cos[x_] :>  (E^(-I*x) + E^(I*x))/2,
   Tan[x_] :> (-I*(-E^(-I*x) + E^(I*x)))/(E^(-I*x) + E^(I*x)),
   Csc[x_] :> (2*I)/(-E^(-I*x) + E^(I*x)),
   Sec[x_] :> 2/(E^(-I*x) + E^(I*x)),
   Cot[x_] :> (I*(E^(-I*x) + E^(I*x)))/(-E^(-I*x) + E^(I*x)),
   Si[x_]  :> -I/2(ExpIntegralE[1, I x] - ExpIntegralE[1, -I x]) + Pi/2,
   Ci[x_]  :> -1/2(ExpIntegralE[1, I x] + ExpIntegralE[1, -I x])
}
TrigToComplexRel = Dispatch[TrigToComplexRel]

TrigToComplex[e_] := e //. TrigToComplexRel

`ComplexToTrigRel = {
   Exp[a_ b_Plus] :> Exp[Expand[a b]],
   Exp[c_Complex x_. + y_.] :> Exp[Re[c] x + y] (Cos[Im[c] x] + I Sin[Im[c] x])
}
ComplexToTrigRel = Dispatch[ComplexToTrigRel]

ComplexToTrig[e_] := Cancel[e //. ComplexToTrigRel]

End[]   (* Algebra`Trigonometry`Private` *)

Protect[ TrigCanonical, TrigExpand, TrigFactor, TrigReduce,
         TrigToComplex, ComplexToTrig ]

EndPackage[]
```

Das Original befindet (befand) sich im Root-Verzeichnis unter "LocalLibrary", "Mathematica", "Packages", "Algebra", "Trigonometry.m" (NeXT).
• L'original se trouve (touvait) dans le directoire Root sous "LocalLibrary", "Mathematica", "Packages", "Algebra", "Trigonometry.m" (NeXT).

■ 7.3.7. Wertelisten, Extraktion von Werten aus Zuweisungsregeln
€ Listes de valeurs, extraction de valeur de règles d'assignation

Studiere die folgende Anwendung:
• Etudie l'application suivante:

```
Needs["Statistics`DescriptiveStatistics`"]

dReport = DispersionReport[{3,4,2,3,6,7,4,8,9,
                            5,2,4}]
```

Das Resultat ist also eine Liste von Regeln! Nun wollen wir einen Wert extrahieren:
• Le résultat est donc une liste de règles! Maintenant nous en extrayons une valeur:

```
MeanDeviation /. dReport
```

Oder wir können das Resultat in eine Variable speichern:
• Oubien nous pouvons mémoriser le résultat dans une variable:

```
x1 = MeanDeviation /. dReport

x1
```

■ 7.3.8. Weiterrechnen mit Werten aus Output in Form von Ersetzungsregeln
€ Continuer de calculer avec des valeurs du output en forme de règles de remplacement

Manchmal erscheint der Output in Form von Ersetzungsregeln.
Mit ihnen kann z.B. wie folgt direkt weitergerechnet werden (Beispiel):
• Parfois l'output apparaît en forme de règles de remplacement. Avec celles-ci on peut continuer de calculer par exemple comme il suit:

```
Remove[x]

wurzeln = Solve[x^2 + 13x - 34 == 0]
```

Hier ist ein solcher Output in Form von Ersetzungsregeln gegeben. Wir verwenden diese Regeln, die nun den Namen "wurzeln" haben, direkt hinter "/." weiter. Wir wollen die Lösungen wieder in die Gleichung einsetzen:
• Voici un tel output en forme de règles de remplacement. Nous continuons d'employer ces règles qui ont dès maintenant le nom "racines" directement derrière "/." . Nous allons remplacer la solution dans l'équation:


```
x^2 + 13x - 34 == 0 /. wurzeln

Simplify[%]
```

Oder: • Ou:

```
x /. wurzeln[[1]]
```

Die erste Regel wurde extrahiert! Oder:
• Cette règle a été extraite! Ou:

```
2x + 13 /. wurzeln[[1]]
```

■ 7.4. Gleichheit
€ Egalité

■ 7.4.1. Allgemeines
€ Généralités

Studiere: • Etudie:

```
?==
```

Anwendung: • Application:

```
Expand[(x + y)^2] == x^2 + 2x y + y^2
```

Vergleich mit "=":
• Comparaison avec "=":

```
Expand[(x + y)^2] =  x^2 + 2x y + y^2
```

Eingebaute Funktionen sind "protected". Man kann dies allerdings ändern. Beispiel:
• Les fonctions incorporées sont "prodected". Mais on peut changer cela. Exemple:

```
?Out

Attributes[Out]
```

Achtung: Das gibt viel Output..... Vgl. 7.1.4.
• Attention: Cela fait beaucoup d'output.... Compare 7.1.4.

```
(*  ??Out   *)

?Unprotect

Unprotect[Out]

Clear[Out]; Protect[Out]
```

Wieviel Output ist es noch?
• Combien d'output cela fait-il encore?

■ 7.4.2. Konvertierung von Gleichheiten in Zuweisungsregeln
€ Convertir des églaités en règles d'assignation

Studiere: • Etudie:

```
?ToRules
```

```
NRoots[x^3 - 5 x^2 - 19 x + 66 == 0 , x]
```

Der Output kommt hier in Form einer logischen oder-Verknüpfpung.
Wir wollen daraus Regeln machen:
• L'output se manifeste ici en forme de composition (avec "ou"). Nous voulons en faire des règles:

```
ToRules[%]
```

■ 7.4.2. Weitere Probleme mit "Gleichheit"
€ Autres problèmes avec "églaité"

Probiere aus: • Essaie:

```
?=
```

```
?==
```

```
?===
```

```
?!=
```

```
?=!=
```

```
?Set
```

```
?Equal
```

```
?SameQ
```

```
?Unequal
```

```
?UnsameQ
```

Studiere: • Etudie:

```
N[Pi, 5] == N[Pi, 30]
```

Hier hat *Mathematica* nur die Gleichheit bis auf die kleinere angegebene
Stellenzahl geprüft.
• Ici *Mathematica* n'a examiné l'égalité que jusqu'au plus petit degré indiqué.

```
Clear[x]; Clear[y];
```

```
x == y
```

Hier gab es nichts zu prüfen.
• Ici il n'y avait rien à examiner.


```
1 != 2
```

Das stimmt. • C'est vraie.

Im folgenden Beispiel wird die syntaktische Gleichheit geprüft, auch wieder bis auf die kleinste angegebene Genauigkeit:
• Dans l'exemple suivant on examine l'égalité syntaxique de nouveau jusqu'à la plus petite exactitude donnée.

```
N[Pi, 3] === N[Pi, 30]
```

Oder: • Ou:

```
N[Pi, 3] =!= N[Pi, 30]
```

```
x =!= y
```

```
x === y
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

7. Probleme zum Thema "Zuordnungen und Regeln"
€ Problèmes au sujet des "applications et règles"

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 7 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 7.
WIR94/99

■ Aufgabe 1 € Problème 1

■ f und g seien verschiedentlich definiert. Finde f[3+a] sowie g[3+a].
€ f et g soient définis différemment. Trouve f[3+a] et g[3+a].

```
Clear[f];Clear[g];
```

a)

```
f[x_]=x/.a->7
```

b)

```
g[x_]:=x/.a->7
```

c)

```
f[3+a]
```

d)

```
g[3+a]
```

e)

```
f[x_]=x + %
```

f)

```
g[x_]:=x + %
```

g)

```
f[x_]=N[x]
```

h)

```
g[x_]:=N[x]
```

i)

```
f[x_]=Expand[2x]
```

j)

```
g[x_]:=Expand[2x]
```

k)

```
?f
```

l)

```
?g
```


■ Aufgabe 2 € Problème 2

■ Probiere aus:
€ Essaie:

```
Integrate[x^2,x]
```

Wieso wohl schreibt *Mathematica* vor die Input line .... := und vor die Output line .... =, ohne den Doppelpunkt?

- Pourquoi est-ce que *Mathematica* écrit devant la ligne Input.....:= et devant la ligne Output ....=, sans le double point?

■ Aufgabe 3 € Problème 3

■ a)

Verdoppelung jeweils des zweiten Elements:

- Redoublement du deuxième élément:

```
{{1,1},{2,2},{3,3}}/.{x_,y_}:>{x,2y}
```

■ b)

Gleiches Vorgehen wie in a), nur mit kleinerer Menge:

- Même procédé que dans a), seulement avec un ensemble plus petite:

```
{{1,1},{2,2}       }/.{x_,y_}:>{x,2y}
```

Wieso hat die Regel jetzt anders funktioniert? (Hinweis: Welche Möglichkeiten bestehen jeweils, das Paar zu interpretieren?)

- Pourquoi la règle a-t-elle fonctionnée autrement? (Indication: Quelles possibilités y a-t-il pour interpréter la paire?)

■ Aufgabe 4 € Problème 4

■ Eigene Regel schreiben:
€ Ecrire une propre règle:

Schreibe eine Regel, die einen Punkt (x, y, z) des 3-dimensionalen Raumes projiziert in den Punkt des 2-dimensionalen Raumes (x, y):

- Ecris une règle, qui projette un point (x, y, z) de l'espace à 3 dimensions dans un point de l'espace à 2 dimensions (x, y):

Regel: • Règle:

```
p[x_,y_,z_]={x,y,z}/.{x_,y_,z_}:>{x,y}
```

Anwendung: • Application:

```
p[7,4,9]
```

■ Aufgabe 5 € Problème 5

■ Eigene Regel schreiben:
€ Ecrire une propre règle:

Schreibe eine Regel, die aus einer Liste von 4 Paaren {xi, yi} und einer Liste von 4 Werten zi eine neue Liste von Tripeln {xi, yi, zi} macht:
• Ecris une règle qui fait d'une liste de 4 paires{xi, yi} et d'une liste de 4 valeurs zi une nouvelle liste de triples {xi, yi, zi}:

Paare: • Paires:

```
paare = Table[{x[i],y[i]},{i,4}]
```

z-Werte: • Valeurs z

```
zWerte = Table[z[i],{i,4}]
```

Noch nicht gewünschte Liste:
• List pas encore désirée:

```
{paare,zWerte}

{paare,zWerte}//MatrixForm

{paare,zWerte}//Transpose//MatrixForm

{paare,zWerte}/.{xyListe_List,zListe_List}:>
Transpose[xyListe] // MatrixForm

{paare,zWerte}/.{xyListe_List,zListe_List}:>
Append[Transpose[xyListe],zListe] // MatrixForm
```

Gewünschte Liste:
• Liste désirée:

```
{paare,zWerte}/.{xyListe_List,zListe_List}:>
Transpose[Append[Transpose[xyListe],zListe]]//
                MatrixForm
```


■ Aufgabe 6 € Problème 6

■ Was geschieht?
€ Que se passe-t-il?

■ (a) Wieso findet trotz der Regel kein Ausmultiplizieren statt?
€ Pourquoi ne se produit-il pas de multiplication malgré la règle?

```
y (x+2)(x-2)/. y z_ -> y Expand[z]
```

■ (b) Verbesserung
€ Correction

Schreibe eine Regel, die das Produkt (x+2)(x-2) ausmultipliziert.
• Ecris une règle qui multiple le produit (x+2)(x-2).

```
y (x+2)(x-2)/. y z_ :> y Expand[z]
```

Hinweis: Was ist der Unterschied bezüglich Parameter bei fixer und bei verzögerter Zuordnung?
• Indication: Quelle est la différence par rapport aux paramètres lors d'une assignation fixe et d'une assignation retardée?

■ Aufgabe 7 € Problème 7

■ Frage:
€ Question:

Ein altes Package enthält die folgende Regel:
• Un vieux Package contient la règle suivante:

```
a_. Cos[x_] Cos[y_] + a_. Sin[x_] Sin[y_] :>
       a Cos[x - y]
```

Wieso enthält es dazu nicht auch die folgende Regel?
• Pourquoi ne contient-il pas non plus la règle suivante?

```
a. Cos[x_ - y_] :>
a Cos[x] Cos[y] + a Sin[x] Sin[y]
```

Hinweise: Unendlicher loop .....
• Indications: Loop infini .....

■ Aufgabe 8 € Problème 8

■ Anwendung auf eine Differentialgleichung:
€ Application pour une équation différentielle:

Verwende DSolve, um die folgende D'Gl. zu lösen: y'=y:
Utilise DSolve pour résoudre l'équation différentielle suivante: y'=y:

```
DSolve[y'[x]==y[x],y[x],x]
```

Probiere, aus der Lösungsliste die Resultatsfunktion herauszugreifen und wieder in die Gleichung einzusetzen, um zu überprüfen, ob das Resultat stimmt!
• Essaie de prender la fonction des résultats dans la liste des solutions et de l'insérer dans l'équation différentielle!

```
r = Flatten[DSolve[y'[x]==y[x],y[x],x]]
```

```
s[x]:=y[x]/.r ; s[x]
```

Einsetzen: • Insérer:

```
D[s[x],x] == s[x]
```

"Putzmaschine" einsetzen
• Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Kurs € Cours

8. Datentypen
€ Types de données

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 8 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 8.
WIR94/99

Mathematica kennt keine Deklarationen von Typen für die Variablen wie z.B. in klassischen Programmiersprachen, z.B. Modula II. Trotzdem.... *Mathematica* kann selber merken, um welchen Typ es sich handelt...
• *Mathematica* ne connaît pas de déclarations de types pour les variables comme p.ex. dans des langues de programmation classiques, p.ex. Modula II. Pourtant.... *Mathematica* s'aperçoit de quel type il s'agit....

■ 8.1. Atomare Typen
€ Types atomiques

■ 8.1.1. Rufe die Information selbst ab
€ Appelle l'information toi-même

Jeder Ausdruck ist aus folgenden "atomaren Typen" zusammengesetzt: Integer, Real, Rational, Complex, Symbol, String.
• Chaque expression est composée des "types atomiques" suivants: Integer, Real, Rational, Complex, String.

```
??Integer
```

```
?Real
```

```
?Rational
```

```
?Complex
```

```
?Symbol
```

```
?String
```

■ 8.1.2. Probiere aus:
€ Essaie:

```
?Head
```

```
Head[3452]
```

```
Head[67.94]
```

```
Head[3 + 4 I]
```

```
Head[E]
```

```
Head[Pi]
```

```
Head["Tschou zämä!"]
```

■ 8.2. Andere Typen und Prädikatenfunktionen
€ Autres types et fonctions de prédicats

■ 8.2.1. Beispiele
€ Exemples

Neben den atomaren Typen gibt es übergeordnete Typen. Z.B. Integer, Real,
Rational u.s.w. sind Zahlen. Ein Vektor ist eine Liste etc.. Um zu prüfen, um welchen Typ es sich gerade handelt, existieren sogenannte Prädikatenfunktionen. Beispiele:
• A part des types atomiques il y a des types supérieurs. P.ex. Integer, Real, Rational etc. sont des nombres. Un vecteur est une liste etc.. Pour examiner de quel type il s'agit, il existe les dites fonctions de prédicats.

```
NumberQ[5/2 + 8 I/5]
```

```
NumberQ[5/2 + 8 I/5 - Pi]
```

```
VectorQ[{a, b, c, d, e, f, g}]
```

■ 8.2.2. Uebersicht
€ Vue d'ensemble

Die Prädikatenfunktionen enden mit "Q". Ihr Wertebereich ist {True, False}. Hier sind sie alle:
• Les fonctions de prédicats se terminent par "Q". Leur domaine de valeurs est "True, False". Voici tous:

```
?*Q
```


■ 8.3. Interne Darstellung in *Mathematica*
€ Représentation intérieure dans *Mathematica*

■ 8.3.1. Allgemeines
€ Généralités

Operationszeichen oder spezielle Symbole wie "+", "/", "#", "/@" u.s.w. werden intern dargestellt durch Funktionen. Mit den Funktionen "FullForm[...]", "FullForm[Hold[...]]" oder "TreeForm" kann die interne Darstellung in *Mathematica* sichtbar gemacht werden.
• Signes d'opérations ao symboled spéciaux tels que "+", "/", "#", "/@" etc. sont représentés intérieurement par des fonctions. Par les fonctions "FullForm[...]", "FullForm[Hold[...]]" ou "TreeForm" on peut rendre visible la représentation intérieure dans *Mathematica*.

```
?FullForm
```

```
?TreeForm
```

Beispiele: • Exemples:

```
FullForm[4/7]
```

```
FullForm[4 - 7 I]
```

```
FullForm[f[x]]
```

```
FullForm[x y^z / w]
```

```
TreeForm[x y^z / w]
```

```
FullForm[2 + 3 15]
```

■ 8.3.2. Erzwungene oder verhinderte Ausführung
€ Exécution obtenue de force ou empêchée

Im letzten Beispiel ist zuerst die Rechnung ausgeföhrt worden. Erst darauf wurde dann die Funktion "FullForm" angewandt. Die gültige Abarbeitungshierarchie kann aber geändert werden. Dazu dienen folgende Befehle:
• Dans le dernier exemple on a d'abord exécuté le calcul. Ensuite seulement on a appliqué la fonction "FullForm". On peut changer la hiérarchie de l'execution valable par les ordres suivants:

```
?Hold
```

```
?HoldForm
```

```
?ReleaseHold
```

```
?Evaluate
```

Beispiele: • Exemples

```
FullForm[Hold[2 + 7 - 4]]
```

```
Hold[2 + 7 - 4]

HoldForm[2 + 7 - 4]

ReleaseHold[Hold[2 + 7 - 4]]

meineListe = {"Or", "And", "Abs"};
Attributes[meineListe]

Attributes[Evaluate[meineListe]]

t[x_] = Table[x^n, {n,4}]

Plot[t[x], {x, -1.5, 1.5}];

Plot[Evaluate[t[x]], {x, -1.5, 1.5}];
```

Was ist passiert?
• Que s'est il passé?

■ 8.3.3. Der "Kopf" eines Ausdrucks und "FullForm"
€ La "tête" d'une expression et "FullForm"

Vergleiche: • Comparer:

```
Head[x + y]

FullForm[x + y]

Head[FullForm[x + y]]

FullForm[{x, y}]

Head[{x, y}]

FullForm[Cos[x^3 - 4 x^2]]

Head[Cos[x^3 - 4 x^2]]
```

■ 8.3.4. Veränderung des "Kopfs"
€ Changement de la "tête"

Vergleiche: • Comparer:

```
FullForm[{x, y, z, w}]

FullForm[x + y + z + w]

{x, y, z, w} /. List -> Plus
```

Was ist passiert? Der "Kopf" ist ausgewechselt worden! Doch *Mathematica* kann noch mehr: Mit "Apply" lässt sich ein weiterer "Kopf" einem Ausdruck voranstellen:
• Que s'est-il passé? On a remplacé la "tête"! *Mathematica* peut cependant davantage. Pour "Apply" on peut placer devant une expression une autre "tête" encore.


```
FullForm[{x, y, z, w}]

summe = Apply[Plus, {x, y, z, w}]

FullForm[summe]

Head[summe]
```

Oder so: • Ou ainsi:

```
Apply[Plus, Range[1000]]
```

Das ist die Summe der natörlichen Zahlen von 1 bis 1000.
• Voici la somme des nombres naturels de 1 à 1000.

■ 8.4. Herauslesen von gewissen Teilen von Ausdrücken
€ Choisir certaines parties d'expressions

■ Allgemeines
€ Généralités

Studiere: • Etudier:

```
?Position

?[[

?Part

neueListe = 2 Range[9]

neueListe[[4]]

Head[neueListe[[4]]]

HoldForm[neueListe[[4]]]
```

Bearbeite: • Travailer:

```
ausdruck = 4 + a - b + Cos[4 Pi x^z - 3]

TreeForm[ausdruck]

ausdruck[[3]]

ausdruck[[4]]

ausdruck[[4, 1]]

ausdruck[[4, 1, 2]]
```

Was ist passiert? --- Studiere weiter die Funktion "Position:
• Que s'est-il passé? --- Continuer d'étudier la fonction "Position":

```
ausdruck1 = Expand[(r + s + w)^3]

Position[ausdruck1,r]

Position[ausdruck1,s]
```

Deute diesen Output! --- Damit kann man auch komponieren:
Interpréter cet output! --- On peut aussi en faire des compositions:

```
u = {ausdruck1[[2]], ausdruck1[[2,2]],
ausdruck1[[2,2,1]], ausdruck1[[5,2]]}

v = Apply[Plus, u]

v /. r->3
```

■ 8.5. Symbole wie "unendlich" etc.
€ Symboles tels que "infini" etc.

■ Allgemeines
€ Généralités

Studiere: • Etudier

```
?Infinity

Head[Infinity]

1 a

0 a

0 5

0 Infinity

?Limit

Limit[1/x, x->0]

Limit[Log[x], x->0]

Head[-Infinity]

FullForm[-Infinity]

Limit[Log[1/x], x->0]

-5 Infinity

7 / 0

?ComplexInfinity

0 / 0
```



```
?Indeterminate
```

■ 8.6. Memory: Wie *Mathematica* speichert
€ Memory: La façon de mémoriser de *Mathematica*

■ 8.6.1. Allgemeines
€ Généralités

Studiere: • Etudier:

```
?MemoryInUse
```

Beispiele: • Exemples:

```
MemoryInUse[]
```

"Putzmaschine" einsetzen:
• Effacer:

```
Remove["Global`@*"]

MemoryInUse[]

t = 1;

MemoryInUse[]
```

Was hat t sowie die ausgeföhrten Operationen "gekostet"?
• Qu'est-ce que t ainsi que les opérations exécutées ont-ils "coûté"?

```
vorher = MemoryInUse[]

Table[Random[],{10000}];

nachher = MemoryInUse[]

N[nachher - vorher] / 1000
```

Wieviel Speicher hat *Mathematica* "verbraten"?
• Combien de mémoire *Mathematica* a-t-il "gaspillé"?

■ 8.6.2. Gemeinsame Speichernutzung (sharing)
€ Utilisation commune de la mémoire (sharing)

Studiere: • Etudier:

```
(   Clear[a, b];
    vorher1 = MemoryInUse[];
    a = Table[Random[],{10000}];
    nachher1 = MemoryInUse[];
    b = a;
    nachher2 = MemoryInUse[];
    {nachher1 - vorher1, nachher2- nachher1}
)
```

Beachte: b hat kein Speicher mehr belegt. b belegt erst dann Speicher, wenn a manipuliert wird. Offenbar gibt es von a und b aus Zeiger auf diesselbe Speicheradresse. Das kann mit der Funktion "Share" auch künstlich bewirkt werden. Im folgenden Beispiel werden 1000 Pseudozufallszahlen im Intervall zwischen 10'000 und 10'100 gerechnet. Da dieses Intervall nur 100 Zahlen enthält, gibt es sicher Duplikate. Mit "Share" kann dabei eine Mehrfachspeicherung vermieden werden. Probiere:

• Retenir: b n'a pas plus occupé de mémoire. b occupe la mémoire seulement si l'on manipule a. Il est évident qu'il a de a et de b des indicateurs pour la même adresse dans la mémoire. On peut obtenir cela aussi artificiellement par la fonction "Share". Dans l'exemple suivant on calcule 1'000 nombres pseudo-aléatoires dans l'intervalle entre 10'000 et 10'1000. Comme cet intervalle ne compte que 100 nombres, il y aura certainement des doubles. Par "Share" on peut éviter une mise en mémoire répétitive. Essaie:

```
?Share
```

```
(   Clear[c];
    vorherSpeicher = MemoryInUse[];
    c = Table[Random[Integer, {10000, 10100}],
                  {10000}];
    nachherSpeicher = MemoryInUse[];
    c = Share[c];
    nachherShare = MemoryInUse[];
    {nachherSpeicher - vorherSpeicher,
     nachherShare- vorherSpeicher}
)
```

Probiere: • Essaie:

```
?ByteCount
```

```
t = 2
```

```
ByteCount[t]
```

```
ByteCount[neueVariable]
```

```
neueVariable = 1;
```

```
ByteCount[neueVariable]
```

```
ByteCount[a]
```

```
ByteCount[{a, b}]
```

Beispiele für Speicherbelegung:

• Exemple pour l'occupation de mémoire:

```
z = 15;  {Head[z], ByteCount[z]}
```

```
z = 1;  {Head[z], ByteCount[z]}
```



```
z = 100;  {Head[z], ByteCount[z]}
```

```
z = 1000000000000000;  {Head[z], ByteCount[z]}
```

```
z = 3.1;  {Head[z], ByteCount[z]}
```

```
z = 3.14159666666666;  {Head[z], ByteCount[z]}
```

```
z = 3/8;  {Head[z], ByteCount[z]}
```

```
z = 3 + 5 I;  {Head[z], ByteCount[z]}
```

```
z = Range[10];  {Head[z], ByteCount[z]}
```

```
z = Range[10000];  {Head[z], ByteCount[z]}
```

```
z = "tschou";  {Head[z], ByteCount[z]}
```

```
z = "t";  {Head[z], ByteCount[z]}
```

```
z = "tschou zaeaeaeaeaeaemmaeaeaeaeaeaeaeae";
    {Head[z], ByteCount[z]}
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

8. Datentypen
€ Types de données

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 8 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 8.
WIR94/99

■ Aufgabe 1 € Problème 1

▪ Untersuche die Interne Darstellung von:
€ Examine la représentation intérieure de

a) 35 a

```
FullForm[35a]
```

b) a 35

```
FullForm[a35]
```

c) - x

```
FullForm[-x]
```

d) - 3

```
FullForm[-3]
```

e) x - 3

```
FullForm[x - 3]
```

f) 3 - x

```
FullForm[3 - x]
```

g) Sqrt[5]


```
FullForm[Sqrt[5]]
```

h) Series[Tan[x],{x,0,9}]

```
Series[Tan[x],{x,0,9}]
```

```
FullForm[Series[Tan[x],{x,0,9}]]
```

g) Normal[Series[Tan[x],{x,0,9}]]

```
Normal[FullForm[Series[Tan[x],{x,0,9}]]]
```

■ Aufgabe 2 € Problème 2

▪ a) Vergleich von (x^y)^z mit x^(y^z)
€ Comparaison de (x^y)^z avec x^(y^z)

x^y^z

```
2^3^4
```

x^(y^z)

```
2^(3^4)
```

(x^y)^z

```
(2^3)^4
```

▪ b) Vergleich der FullForm
€ Comparaison de FullForm

x^y^z

```
2^3^4
```

```
FullForm[2^3^4]
```

x^(y^z)

```
2^(3^4)
```

```
FullForm[2^(3^4)]
```

(x^y)^z

```
(2^3)^4
```

```
FullForm[(2^3)^4]
```

■ Aufgabe 3 € Problème 3

■ Untersuchung der Memory-Belegung bei Operationen € Examiner l'occutation de la mémoire lors d'opérations

Operation:
Liste von 1000 Integer-Zufallszahlen zwischen 100 und 200 rechnen:
• Opération:
Calculer une liste de 1000 nombres aléatoires "Integer" entre 100 et 200:

```
aufruf1 = MemoryInUse[];
kleineInt = Table[Random[Integer,{100,200}],
                  {1000}];

aufruf2 = MemoryInUse[];
aufruf2 - aufruf1
```

Operation:
Liste von 1'000 Integer-Zufallszahlen zwischen 300 und 400 rechnen:
• Opération:
Calculer une liste de 1'000 nombres aléatoires integer entre 300 et 400:

```
kleineInt = Table[Random[Integer,{300,400}],
                  {1000}];

aufruf3 = MemoryInUse[];
aufruf3 - aufruf2
```

Wieviel Memory ist mehr benutzt worden?
(*Mathematica* ist so gebaut worden, weil eine Klasse mehr benutzt wird...)
• Combien de mémoire en plus a-t-on occupé?
(*Mathematica* a été construit ainsi, parce qu'on utilise une classe...)

Operation:
Liste von 1'000 Real-Zufallszahlen (default-Präzision) rechnen:
• Opération:
Calculer une liste de 1'000 nombres aléatoires réels (précision default):

```
realInt = Table[Random[],{1000}];

aufruf4 = MemoryInUse[];
aufruf4 - aufruf3
```


■ Aufgabe 4 € Problème 4

■ Was ist die grösste speicherbare quadratische Matrix bei 4 Megabytes? € Quelle est la plus grande matrice carrée qu'on peut mémoriser?

Annahme: Zur Speicherung einer Floating-point-Zahl braucht es 32 = 2^5 Bytes.
• Supposition: Pour mémoriser un nombre à virgule flottante on utilise 32 = 2^5 bytes.

```
NSolve[2^x==10^6,x]
```

1 Mb entspricht etwa 2^20 Bytes.
• 1 Mb correspond à environ 2^20 bytes.

Anzahl Floating-point Zahlen in 4 Mb:
• Nombre (quantité) de nombres à virgule flottante dans 4 Mb:

```
a = 4 2^19.9316/2^5
```

Seitenlänge der quadratischen Matrix:
• Longeur d'une côté de la matrice carrée:

```
s = Floor[Sqrt[a]]
```

■ "Putzmaschine" einsetzen € Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

9. Funktionen definieren
€ Définir des fonctions

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 9 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 9.
WIR94/99

■ 9.1. Einfache Funktionen
€ Fonctions simples

■ 9.1.1. Funktionen selber schreiben: Falls in *Mathematica* die gewollte Funktion nicht schon vorhanden ist, kann man versuchen, sie selber zu schreiben.
€ Ecrire les fonctions soi-même: Si la fonction voulue n'est pas contenue dans *Mathematica*, on peut l'écrire soi-même.

Erst "Clear", damit keine Ueberraschung auftritt! Beispiel: Quadrieren...
• D'abord "Clear", pour ne pas avoir de susrprise! Exemple: Elever au carré...

```
Clear[quadrat];
quadrat[x]:=x^2;
quadrat[x]

{quadrat[y], quadrat[5], quadrat[12]}
```

Offenbar klappt es bei dieser Funktion nicht mit demVariablenwechsel resp. mit dem Einsetzen von Werten. Wieso?
• Il est évident que le changement de variables ne fonctionne pas pour cette fonction le placement de valeurs non plus. Pourquoi?

```
??quadrat

?_
```

Die Variable x ist kein "Muster", das för irgend einen andern Ausdruck oder Wert stehen kann. Nehmen wir "x_" statt "x", so klappt es:
• La variable x n'est pas un "patron" qui peut remplacer une expression ou une valeur quelconque. Si nous prenons "x_" au lieu de "x", cela fonctionne:


```
Clear[quadrat1];
quadrat1[x_]:=x^2;
quadrat1[x]

{quadrat1[y], quadrat1[5], quadrat1[12]}

??quadrat1
```

Oder: • Ou:

```
Table[quadrat1[n],{n,12}]
```

Oder: • Ou:

```
quadrat1[9z^3+2z-1]

quadrat1[9z^3+2z-1] // Expand
```

Oder: • Ou:

■ 9.1.2. Funktionen mit mehreren Variablen:
€ Fonction avec plusieurs variables:

Ein Beispiel, dass alles sagt (Cosinus-Satz):
• Un exemple qui dit tout (théorème du cosinus):

```
Clear[c];
c[a_, b_, gamma_] = Sqrt[a^2 + b^2
                    - 2 a b Cos[gamma]];
c[u, v, w]

c[3, 5, Pi/3]
```

■ 9.1.3. Funktionen definiert in mehreren Schritten:
€ Fonction définies en plusieurs étapes:

Beispiel: Umrechnung in Polarkoordinaten:
• Exemple: Transformation en coordonnées polaires:

```
PolarTransformation[r_, phi_] := (
           x = r Cos[phi];
           y = r Sin[phi];
           {x, y}              );
PolarTransformation[a, b]
```

Neudefinition: Die Umrechnung in Polarkoordinaten könnte auch für einen Vektor aufgerufen werden. Daher ist eine zweite Definition ratsam:
• Nouvelle définition: La transformation en coordonnées polaires pourrait être appelée aussi pour un vecteur. C'est pourquoi on conseille un deuxième définition:

```
PolarTransformation[{r_, phi_}] :=
               PolarTransformation[r, phi];
PolarTransformation[{a, b}]
```

```
PolarTransformation[{3, 2}]

PolarTransformation[3, 2]

PolarTransformation[3, 2] // N
```

■ 9.2. Typenprüfung
€ Examen de types

■ 9.2.1. f[arg_typ] € f[arg_typ]

"f[arg_typ]" funktioniert nur, falls gilt "Head[arg] === typ". Probiere:
•
"f[arg_typ]" fonctionne seulement s'il vaut "Head[arg] === typ". Essaie:

```
?Binomial

Clear[pascal];
pascal[n_Integer]:=Table[Binomial[n,i],{i,0,n}];

pascal[0]

pascal[1]

pascal[2]

pascal[3]

pascal[4]

pascal[5]

pascal[5.6]

pascal[6/5]
```

Ein anderes Beispiel (ausprobieren):
• Autre exemple (essayer):

```
Clear[mittelWert];

mittelWert[x_List] := Apply[Plus, x]/Length[x];
mittelWert[{1,2,3,4,5}]

mittelWert[1,2,3,4,5]
```

Test, ob ein Ausdruck ein Polynom, eine Primzahl, ein Vektor u.s.w. ist, d.h. ob ein Ausdruck unter einer Prädikatenfunktion "wahr" ergibt. Beispiel:
• Tester si une expression est un polynôme, un nombre premier, un vecteur etc., c'est-à-dire, si une expression est "vraie" sous une fonction de prédicats. Exemple:

```
?*Q

{PrimeQ[45], PrimeQ[17]}
```


Anwendung auf die Definition einer Funktion:
• Application à la définition d'une fonction:

```
Clear[halb];
halb[x_?EvenQ] := x/2;
{halb[1], halb[2], halb[3], halb[4], halb[5]}

{halb[2], halb[4], halb[6], halb[8], halb[9]}
```

Was ist passiert?
• Que s'est-il passé?

Ein Beispiel in mehreren Schritten:
• Un exemple en plusieurs étapes:

```
Remove[signum];
signum[0                    ] :=  0;
signum[x_/; Positive[x]==True ] :=  1;
signum[x_/; Negative[x]==True ] := -1;

{signum[3], signum[-1], signum[0], signum[2]}

??Positive

??Integer

Positive[1]

signum[1]

??signum
```

■ 9.3. Bedingte Ausführung
€ Exécution conditionnée

■ Betrachte das obige Beispiel mit "signum"!
€ Considère l'exemple ci-dessus avec "signum"!

Bei der Definition von "signum" ist in der Klammer "[ ]" nach "x_" das Symbol "/;" angegeben. Darauf folgt jeweils eine Bedingung! Damit können Funktionen punktweise oder intervallweise definiert werden!!!!!! Anwendungen:
• Dans la définition de "signum", le symbole "/;" est indiqué dans la parenthèse "[ ]" après "x_". Il suit chaque fois une condition! Ainsi on peut définir des fonctions ponctuellement ou par intervalles! Applications:

```
Plot[signum[x], {x, -3, 3}];
```

Oder: • Ou:

```
Clear[rechteck];
rechteck[x_/; (-1<=x && x<=1) ]:= 2;
rechteck[x_/; Abs[x] > 1      ]:= 0;
Plot[rechteck[x], {x, -3, 3}];
```

Etwas "schöner":
• Un peu plus esthétiquement:

```
Plot[rechteck[x], {x, -3, 3},
    PlotStyle -> {{Thickness[0.01],
                   GrayLevel[0.4]}}];
```

Solche Funktionen können sogar integriert werden:
• On peut même intégrer de telles fonctions:

```
??rechteck

NIntegrate[rechteck[x], {x, -1, 1}]

NIntegrate[rechteck[x], {x, -3, 3}]
```

Was ist passiert? Studiere:
• Que s'est-il passé? Etudie:

```
?Which

Remove[funkt];
funkt[x_] := Which[x<=Pi,-2 Sin[x],x>Pi,
            (x^1.3-Pi^1.3) Cos[x]];
Plot[funkt[x],{x,-2Pi,3Pi}];

Remove[funkt];
funkt[x_] := Which[x<=Pi,2 Sin[x],x>Pi,
            (x^1.3-Pi^1.3) Cos[x]];
Plot[funkt[x],{x,-2Pi,3Pi}];
```

■ 9.4. Vorgegebene Werte
€ Valeurs connées

■ 9.4.1. Das Muster "x_.":
€ Le patron ("Pattern") "x_.":

Das Muster "x_." bewirkt, dass x "default" mit dem neutralen Element der am
"stärksten" mit x verknüpften Operation belegt wird. Dazu ein Beispiel:
• Le patron "x_." fait que x "default" est occupé par l'élément neutre de l'opération qui est le plus "fortement" lié à x.
Voici un exemple:

```
Clear[f];
f[x_. + y_. z_.^e_.] := {x, y, z, e};


Clear[f];
f[x_. + y_. z_^e_.] := {x, y, z, e};


?f

f[4]
```



```
{f[0], f[1], f[2], f[3], f[4], f[5]}

{x, y, z, e}
```

Finde heraus, welche Werte "default" jeweils für x, y, z und e genommen worden sind und wieso!
• Cherche quelles valeurs "default" ont été prises respectivement pour x, y, z et e et pour quelle raison!

■ 9.4.2. Das Muster "x_:" :
€ Le patron "x_:" :

Das Muster "x_:Wert" bewirkt, dass x "default" mit Wert belegt wird. Dazu ein Beispiel:
• Le patron "x_:Valeur" fait que x "default" est occupé par valeur. Voici un exemple:

```
Clear[UnserBereich];
UnserBereich[start_:34, ziel_] :=
                    Range[start, ziel];
UnserBereich[100]

UnserBereich[80, 100]

UnserBereich[36]

UnserBereich[34]

UnserBereich[30]
```

■ 9.5. Die Abarbeitungshierarchie bei Regeln

Mathematica bearbeitet die spezielleren Regeln vor den generelleren Regeln. Mit "?" kann die Abarbeitungshierarchie eingesehen werden.
• *Mathematica* travaille les règles spéciales avant les règles générales. Avec "?" on peut voir la hiérarchie de travail.

```
Clear[r];
r[x_]       := x^2;
r[x_Integer]:= x;
r[5]        := 11;
?r
```

Beachte, dass die Reihenfolge der Ausgabe nicht mit der der Eingabe übereinstimmt. Wieso wohl?
Considère que l'ordre de la sortie ne correspond pas à celui de l'entrée. Pourquoi?

```
{r[5], r[5.5], r[6], r[r.5], r[7]}
```

■ 9.6. Zusammenfügen von Regeln mit "f/: "
€ Assemblage de règles par "f/: "

■ Operationen links vom Zeichen ":= "
€ Opérations à gauche du signe ":= "

Z.B. f[x_] + g[x_] := h[x_] "funktioniert" nicht. Beispiel:
• P.ex. f[x_] + g[x_] := h[x_] ne "fonctionne" pas. Exemple:

```
f[x_] + g[x_] := h[x_]
```

Wie lässt sich das Funktionieren trotzdem erzwingen? -
Dafür ist das Symbol "f/ : " vorgesehen. Studiere das folgende Beispiel:
• Comment obtenir quand même le fonctionnement? -
Pour cela est prévu le symbole "f/ : ". Etudie l'exemple suivant:

```
Remove[f];
f/: f[x_] + g[x_] := h[x_]

?f

f[5] + g[5]
```

f wird hier assoziiert mit f[x]+g[x] := h[x]. Wird f[x]+g[x] gerufen, so erscheint h[x]. Andernfalls erscheint f[x] oder f[x1]+g[x2].
• Ici on associe f avec f[x]+g[x] := h[x]. Si on appelle f[x]+g[x], il arrive h[x]. Autrement il apparaît f[x] ou f[x1]+g[x2].

```
f[5]

f[5] + g[6]
```

Beispiel: • Exemple:

```
Remove[f];
f/: f[x_] + Sin[x_] := x^3;
f[3] + Sin[3]
```

Vergleiche: • Compare:

```
??_

?_.

?:

?/;

???
```


■ 9.7. Dokumentieren der eigenen Funktionen
€ Documenter les propres fonctions

■ 9.7.1. Verschiedenes Verhalten einer Transformationsregel:
€ Comparement différent d'une règle de transformation

Beachte, dass die Regel s[-x_] :> -s[x] den Ausdruck s[-a] in -s[a] transformiert, s[-3] aber nicht transformiert. Was wird mit s[1-x] geschehen und wieso?
• Considère que la règle s[-x_] :> -s[x] transforme l'expression s[-a] en -s[a], mais ne transforme pas s[-3]. Que se passe-t-il avec s[1-x] et pourquoi?

```
sRegel = s[-x_] :> -s[x]

s[-a] /. sRegel

s[-3] /. sRegel
```

Wieso geschieht das? Beobachte, wie *Mathematica* -a und -3 interpretiert:
• Pourquoi cela arrive-t-il? Observe comment *Mathematica* interprète -a et -3:

```
FullForm[-a]

FullForm[-3]
```

Der Ausdruck -a ist mit dem Muster -(x_) verträglich, der Ausdruck -3 nicht.
• L'expression -a est compatible avec le patron -(x_), l'expression -3 ne l'est pas.

```
FullForm[1-x]
```

Der Ausdruck (1-x) ist mit dem Muster -(x_) nicht verträglich:
• L'expression (1-x) n'est pas compatible avec le patron -(x_).

```
s[1-x] /. sRegel
```

■ 9.7.2. Der Ausdruck ' Funktion ::usage = "Erklärung" '
€ L'expression ' Funktion ::usage = "explication" '

Bisher ist öfters schon das Symbol "?" erschienen. Damit ist einige Information abrufbar. Beispiel:
• Jusqu'à présent le symbole "?" est apparu plusieurs fois. Par cela on peut appeler bien l'information. Exemple:

```
?f

?Sin

??Sin
```

Manchmal möchte man aber gern dem System eigene Kommentare einverleiben, die dann wieder abgerufen werden können. Das ist möglicn mit "...::usage=...".

Beispiel: Wir verwenden die oben definierte Funktion "pascal".
• Parfois on aimerait incorporer au système ses propres commentaires, qu'on pourrait de nouveau appeler. Cela est possible avec "...::usage=...".
Exemple: Nous employons la fonction "pascal" définie ci-dessus.

```
Clear[pascal];
pascal[n_Integer]:=Table[Binomial[n,i],{i,0,n}];
```

```
?pascal
```

Wir fügen nun an:
• Nous ajoutons:

```
pascal::usage = "pascal[n] gibt die n-te Zeile
des Pascalschen Dreiecks aus - sort la ligne n
du triangle de Pascal."
```

Abfragen: • Appeler:

```
?pascal
```

```
??pascal
```

Gute Sache, nicht?
• C'est bon, n'est-ce pas?

■ 9.8. Attribute
€ Attributs

■ Spezielle Funktionen bestimmen:
€ Déterminer des fonctions spéciales:

Funktionen haben Attribute. Beispiel "Sinus":
• Les fonctions ont des attributs. Exemple "Sinus":

```
Attributes[Sin]
```

```
??Attributes
```

```
??Attributes
```

Die möglichen Attribute sind vom System vorgegeben. Einige Beispiele:
• Les attributs possibles sont donnés par le système. Quelques exemples:

```
??Constant
```

```
?Flat
```

```
?Hold*
```

```
?HoldAll
```

```
?HoldFirst
```



```
?HoldRest
```

```
?Listable
```

```
?Locked
```

```
?Orderless
```

```
?Protected
```

```
?ReadProtected
```

■ 9.9. Die Art der Abarbeitung eines Ausdrucks
€ La façon de travailler une expression

■ 9.9.1. Normalfall
€ Cas normal

Folgende Reihenfolge ist im System voreingestellt: Zuerst Kopf des Ausdrucks abarbeiten. Dann Elemente (Teile) des Ausdrucks abarbeiten.
Wenn ein Element kein Atomarer Ausdruck ist: Teile des Elements abarbeiten etc.. Dieses Procedere wird rekursiv angewandt.
• L'ordre suivant est déjà réglé dans le système: Travailler d'abord l'en-tête de l'expression. Travailler ensuite les éléments (parties) de l'expression. Si un élément n'est pas une expression atomique: Procéder par parties etc.. Ce procédé est appliqué récursivement.

■ 9.9.2. Ausnahmefall
€ Exception

Wenn ein Element oder Ausdruck das Attribut HoldFirst, HoldAll, HoldRest hat oder die Funktion Evaluate aufruft, wird die Voreinstellung durchbrochen. Erkunde, wo solche Attribute vorkommen:
• Quand un élément en une expression a l'attribut HoldFirst, HoldAll, HoldRest ou appellent la fonction Evaluate, le règlement préalable est interrompu. Cherch où se trouvent tels attributs:

```
??Sin
```

```
??Plot
```

```
??Table
```

```
??Attributes
```

```
??Evaluate
```

Vergleiche die Resultate bei Abänderung der Abarbeitungsreihenfolge und erkläre den Unterschied der Resultate:
• Compare les résultats lors des changements de l'ordre de travail et explique la différence des résultats:

```
Table[Random[], {5}]
```

```
Table[Evaluate[Random[]], {5}]
```

Wieso kommt das so, wie es kommt?
• Pourquoi cela se déroule ainsi?

■ 9.10. Diskrete Funktionen
€ Fonctions discrètes

Ein Beispiel:
• Un exemple:

```
Clear[ungerade, t];
ungerade[x_?OddQ ] := (ungerade[x] = 1);
ungerade[x_?EvenQ] := (ungerade[x] = -1);


t = Table[ungerade[x], {x,1, 15}]

ListPlot[t, PlotStyle -> PointSize[0.04]];

?ListPlot
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Uebungen € Exercices

9. Funktionen definieren
€ Définir des fonctions

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 9 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 9.
WIR94/99

■ Aufgabe 1 € Problème 1

■ Untersuche, was die unten definierte Funktion f macht:
€ Examine ce que fait la fonction f définie ci-dessous:

Definition: • Définition:

```
f[2]  = 3    ;
f[u] := 2 u^2;
f[v_]:= 1/v  ;
u     = 2;
```

Was ist f[u] ?
• Qu'est-ce f[u] ?

```
f[u]
```

Was ist mit f passiert? Untersuche die möglichen Graphen!
Que s'est-il passé? Examine les graphes possibles!

```
Plot[f[x],{x,0.5,6}];

Plot[f[u],{x,0.5,6}];

f[2]
```

Was ist passiert?
• Que s'est-il passé?

■ Aufgabe 2 € Problème 2

■ **Definiere eine Funktion mitte[x], die das Integer- Resultat ausgibt für ungerade Werte von x!**
€ Définis une fonction mitte[x], qui sort le résultat Integer pour les valeurs impairs de x!

Definition: • Définition:

```
Clear[f];
f[x_?OddQ] := (x+1)/2
```

Ausprobieren: • Essayer:

```
f[3.2]
```

```
f[3]
```

```
f[2]
```

Was ist passiert? • Que s'est-il passé?

■ Aufgabe 3 € Problème 3

■ **Definiere eine Funktion, die als Argument ein Zahlenpaar {a, b} hat und als Wert das Paar {a, 2b} ausgibt. Wende die Funktion an auf die Liste {{1, 1}, {2, 2}}:**
€ Définis une fonction, qui a pour argument une paire de nombres {a, b} et qui sort comme valeur la paire {a, 2b}. Applique la fonction à la liste {{1, 1}, {2, 2}}:

Wende mit "Map" die Funktion an auf die Liste {{1, 1}, {2, 2}}.
• Applique par "Map" la fonction à la liste {{1, 1}, {2, 2}}.

```
Clear[verdoppelnY];
verdoppelnY[{x_, y_}]:={x, 2y}
```

```
Map[verdoppelnY, {{1, 1},{2, 2}}]
```

Was macht "Map" hier?
• Que fait "Map" ici?

■ Aufgabe 4 € Problème 4

■ **Definiere eine Funktion "dreieck[x]" wie folgt:**
€ Eéfinis une fonction "dreieck[x]"

dreieck[x] ist gleich 1 - Abs[x] für x<1. Sonst ist der Funktionswert null.
Mache einen Plot!
• dreieck[x] est égal 1 - Abs[x] pour x<1. Autrement la valeur de la fonction est zéro. Fais un plot!

```
Clear[dreieck];
dreieck[x_/; (-1<=x && x<=1)]:= 1 - Abs[x];
dreieck[x_/; Abs[x]>1]:= 0;
Plot[dreieck[x],{x,-3,3},PlotStyle -> {{
    Thickness[0.01],GrayLevel[0.5]}}];
```

■ Aufgabe 5 € Problème 5

■ **Definiere eine Funktion "signum[x]" wie folgt:**
€ Définis une fonction "signum[x]" comme il suit:

signum[x] ist gleich 1 für x>0, -1 für x<0 und 0 für x=0. Mache einen Plot! (Definiere dazu signum für Floating-Point-Zahlen 0.0!) Schreibe einen Kommentar für die Dokumentation!
• signum[x] est égal 1 pour x>0, -1 pour x<0 et 0 pour x=0. Fais un plot. (Définis signum pour les nombres à flottante 0.0!) Ecris un commentaire pour la documentation!

```
Clear[signum];
signum[0] = 0        ;
signum[x_/; x>0]:= 1 ;
signum[x_/; x<0]:= -1;
signum::usage="signum[x] ist gleich 1 für
positive x, -1 für negative x und 0 für
x = 0 - est égal 1 pour x positif, -1 pour
x négatif et 0 pour x = 0.";
Plot[signum[x],{x,-3,3},PlotStyle -> {{
    Thickness[0.01],GrayLevel[0.5]}}];
```

```
Clear[signum];
signum[0.] = 0.      ;   (*Geändert!*)
signum[x_/; x>0]:= 1 ;
signum[x_/; x<0]:= -1;
signum::usage="signum[x] ist gleich 1 für
positive x, -1 für negative x und 0 für
x = 0 - est égal 1 pour x positif, -1 pour
x négatif et 0 pour x = 0.";
Plot[signum[x],{x,-3,3},PlotStyle -> {{
    Thickness[0.01],GrayLevel[0.5]}}];
```

```
?signum
```

```
??signum
```

■ Aufgabe 6 € Problème 6

■ Definiere die Funktion "myRange" wie folgt:
€ Définis la fonction "myRange" comme il suit:

myRange[x] gibt für ein positives Integer-Argument n die Zahlen von 0 bis n aus. Für zwei Argumente m und n mit "n < m" gibt die Funktion alle natürlichen Zahlen von n bis m aus. Für drei Argumente n, m und d gibt sie die Zahlen n, n+d, n+2d, ... , n+kd aus mit n+kd<m.
• myRange[x] sort pour un argument Integer positif n les nombres de 0 à n. Pour deux arguments m et n avec "n < m", la fonction sort tous les nombres naturels de n à m. Pour trois arguments n, m et d, elle sort les nombres n, n+d, n+2d, ... , n+kd avec n+kd<m.

```
Clear[myRange];
myRange[n_?Integer,m_?Integer,d_?Integer/;
{n>=0 && m>=n && d>=0}]:= Range[n,m,d];
myRange[n_?Integer,m_?Integer /;
{n>=0 && m>=n}]:= Range[n,m];
myRange[start_:0, n_?Integer /; n>=0]:=
Range[start,n];
Print[{6,Range[6]}];
Print[{-6,Range[-6]}];
Print[{0,Range[0]}];
Print[{{5,11},Range[{5,11}]}];
Print[{5,11,Range[5,11]}];
Print[{5,-3,Range[5,-3]}];
Print[{5,3,Range[5,3]}];
Print[{5,18,4,Range[5,18,4]}];
```

■ Aufgabe 7 € Problème 7

■ Verschiedenes Verhalten einer Transformationsregel:
€ Comportement différent d'une règle de transformation:

Beachte, dass die Regel "s[-x_] :> -s[x]" den Ausdruck "s[-a]" in "-s[a]" transformiert, "s[-3]" aber nicht transformiert. Was wird mit "s[1-x]" geschehen und wieso?
• Considère que la règle "s[-x_] :> -s[x]" transforme l'expression "s[-a]" en "-s[a]", mais ne transforme pas "s[-3]". Que se passe-t-il avec "s[1-x]" et pourquoi?

```
sRegel = s[-x_] :> -s[x]

s[-a] /. sRegel

s[-3] /. sRegel
```

Wieso geschieht das? Beobachte, wie *Mathematica* -a und -3 interpretiert:
• Pourquoi cela arrive-t-il? Observe, comme *Mathematica* interprète -a et -3:

```
FullForm[-a]
```



```
FullForm[-3]
```

Der Ausdruck -a ist mit dem Muster -(x_) verträglich, der Ausdruck -3 nicht.
• L'expression -a est compatible avec le patron -(x_) , l'expression -3 ne l'est pas.

```
FullForm[1-x]
```

Der Ausdruck (1-x) ist mit dem Muster -(x_) nicht verträglich:
• L'expression (1-x) n'est pas compatible avec le patron -(x_):

```
s[1-x] /. sRegel
```

■ Aufgabe 8 € Problème 8

■ Approximative Berechnung von Pi/4 durch eine rekursive Funktion:
€ Calcul approximatif de Pi/4 par une fonction récursive:

Definiere die rekursive Funktion piDurchVier[n_] um approximativ Pi/4 zu berechnen wie folgt:
• Définis la fonction récursive piDurchVier[n_] pour calculer approximativement Pi/4 comme il suit:

Pi/4 ist etwa: • Pi/4 est environ:

2*4*4*6*6*8*8*10*10...2n / (3*3*5*5*7*7*9*9*...(2n+1)).

Entwickle: • Développe:

piDurchVier[1] = 2/3
piDurchVier[2] = 2*4*4/(3*3*5) = piDurchVier[1]*4*4/(3*5)
piDurchVier[3] = 2*4*4*6*6/(3*3*5*5*7)
= piDurchVier[2]*6*6/(5*7)
piDurchVier[4] = piDurchVier[3]*8*8/(7*9)

```
RemoveAll[piDurchVier];
piDurchVier[1] = 2/3;
piDurchVier[n_] :=
    piDurchVier[n-1] (2n)^2 / ((2n-1)(2n+1));
??piDurchVier

Table[piDurchVier[i],{i,5}]

Table[piDurchVier[i],{i,5}] // N
```

Kontrolle: • Contrôle:

```
N[Pi/4,10]

N[piDurchVier[200]]
```

Was hälst Du von der Sache?
• Qu'en penses-tu?

■ Aufgabe 9 € Problème 9

■ Spezielle Funktionen bestimmen:
€ Déterminer des fonctions spéciaales:

Bestimme die Funktionen, deren Namen mit den Buchstaben A - E beginnen und das Attribut "Listable" haben.
Hinweis: Was tun die Funktionen "Names", "MemberQ", "Attributes" und "Select"?
• Détermine les fonctions dont le nom commence par les lettres A - E et qui ont l'attribut "Listable".
Indication: Que font les fonctions "Names", "MembreQ", "Attributes" et "Select"?

```
??Names
```

```
??MemberQ
```

```
??Attributes
```

```
??Select
```

Suche alle "Listablen" Funktionen:
• Cherche toutes les fonctions "Listables":

```
RemoveAll[listbareQ];
listbareQ[symbol_String]:=
    MemberQ[Attributes[symbol], Listable]
```

Was tun die Teile?
• Qu'en font les parties?

```
Attributes["Cos"]
```

```
MemberQ[Attributes["Cos"], Listable]
```

```
listbareQ["Cos"]
```

Versuchen wir, die Funktionen zu bestimmen:
• Essayons de déterminer les fonctions:

```
Select[Names["A* B* C* D* E*"], listbareQ]
```

```
Select[Names["A*"], listbareQ]
```

```
Select[Names["B*"], listbareQ]
```

```
Select[Names["F*S*"], listbareQ]
```

```
sN[x_]:=Select[Names[x], listbareQ]
```

```
sN["A*"]
```

```
Union[sN["A*"],sN["B*"],sN["C*"],sN["D*"],
                sN["E*"]]
```


■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

10. Lokale Variablen und prozedurales Programmieren
€ Variables locales et programmation procédurale

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 10 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 10.
WIR94/99

■ 10.1. Globale Variablen
€ Variables globales

Mathematica stört es nicht, dass im folgenden Beispiel zweimal dieselbe globalen Variable (hier "i") in verschiedenen Bedeutungen gebraucht wird:
• Cela ne dérange pas *Mathematica* que dans l'exemple suivant on emploie deux fois la même variable globale (ici "i") avec deux significations différentes:

```
f[n_]:=Table[g[i],  {i,n}];
g[m_]:=Table[Log[i],{i,m}];
```

```
f[3]
```

```
??i
```

Zum Vergleich: • Pour comparer:

```
f[n_]:=Table[g[i],  {i,n}];
g[m_]:=Table[Log[j],{j,m}];
```

```
f[3]
```

■ 10.2. Lokale Variablen im Unterschied zu globalen Variablen

€ Variables locales à la différence de variables globales

■ 10.2.1. Lokale Variablen in einer "Block"-Prozedur
€ Variables locales dans une procédure "Block"

```
??Block
```

Betrachte das folgende Beispiel:
• Considère l'exemple suivant:

```
Clear[f,g];
f[n_] := Block[{i}, Table[  g[i], {i, n}]];
g[m_] := Block[{i}, Table[Log[i], {i, m}]]
```

Ausprobieren: • Essayer:

```
i
```

```
?f
```

```
f[n]
```

```
f[3]
```

```
f[5]
```

```
i
```

Was ist passiert? (Welcher Wert ist in "i" gespeichert?) Betrachte:
• Aue s'est-il passé? (Quelle valeur est mémorisée dans "i"?) Considère:

```
Clear[f,];
Block[{i}, i]
```

■ 10.2.2. Lokale Variablen in einer "Module"-Prozedur
€ Variables locales dans une procédure "Module"

```
??Module
```

Vergleiche mit dem letzten Beispiel:
• Compare au dernier exemple:

```
Clear[f,];
Module[{i}, i]
```

Was bedeutet wohl die ausgegebene Zahl?
• Que signifie le nombre sorti?

Betrachte das folgende Beispiel:
• Considère l'exemple suivant:


```
Clear[f,g];
f[n_] := Module[{i}, Table[  g[i], {i, n}]];
g[m_] := Module[{i}, Table[Log[i], {i, m}]]
```

Ausprobieren: • Essayer:

```
?f
```

```
f[n]
```

```
f[3]
```

```
f[5]
```

```
i
```

Was ist passiert? (Welcher Wert ist in "i" gespeichert?)
• Que s'est-il passé? (Quelle valeur est mémorisée dans "i"?)

■ 10.2.3. Lokale Variablen mit "With"
€ Variables locales avec "With"

Damit lassen sich Ausdrücke manipulieren:
• Par cela on peut manipuler des expressions:

```
?With
```

```
With[{x=2, y=a, z=b^3}, y Log[x^z]]
```

```
Clear[x,y,z,r,s];
Solve[{x+2==r-s,s+1==y+2z,2r+z==-4x},{x,r}]
```

```
With[{z=4, s=Sin[z^2]},
Solve[{x+2==r-s,s+1==y+2z,2r+z==-4x},{x,r}]]
```

■ 10.3. Prozedurale Programmierung
€ Programmation procédurale

Mathematica erlaubt prozedurale Programmierung, d.h. Befehlssequenzen mit Wiederholungen und Verzweigungen. Studiere die folgenden Befehle:
• *Mathematica* permet la programmation procédurale, c'est-à-dire des séquences d'ordres avec des répéditions et des ramifications. Etudier les ordres suivants:

```
??Do
```

```
?For
```

```
?While
```

```
?If
```

```
?Which
```

```
?Switch
```

■ 10.3.1. Arithmetische Operationen auf bestehenden Variablen wie Inkrementierung u.s.w. € Opérations arithmétiques sur des variables existantes comme "increment" etc.

Studiere die folgenden Beispiele:
• Etudie les exemples suivants:

```
??++

?+=

?-=

?*=

?/=

?DivideBy

i=1

++i

i

i++

i

i

i +=6

i -=5

i *=7

i /=4
```

■ 10.3.2. Iterative Konstruktionen € Constructions itératives

Studiere die folgenden Beispiele:
• Etudie les exemples suivants:

```
Do[Print[3 n^2], {n,7}]

For[n=1, n<=7, ++n, Print[3 n^2]]

n=1;
While[n<=7, (Print[3 n^2]; n++)]
```


■ 10.3.3. Logik € Logique

Studiere die folgenden Befehle:
• Etudie les ordres suivants:

```
?&&

?And

?||

?Or

?!

?Not

?Xor
```

Studiere die folgenden Beispiele:
• Etudie les exemples suivants:

```
(1 < 2) || (4 > 5/0)

(4 > 5/0) || (1 < 2)
```

Von wo her wird also geklammert?
• D'où viennent les parenthèses?

■ 10.3.4. Wahrheitstabellen € Tableaux de verité

Ein Beispiel: Die folgende Aussageform soll untersucht werden:
• Un exemple: La forme propositionelle suivante soit examinée:

```
Clear[f];
f[x_:0, y_:0, z_:0, w_:0, u_:0, v_:0]:=
       !(!x || (x && y)) || (x || ! y) && w;
AppendTo[Attributes[f], Listable]

??f
```

Die nachstehende Funktion definiert den "Input" in die Wahrheitstabelle:
• La fonction ci-dessous définit l'"Input" dans le tableau de vérité:

```
Remove[tabteil,tabelle];
tabteil[n_, k_]:= Permutations[
               Table[Ceiling[Floor[n/i]/k],
                        {i,1,k}]];
tabelle = {};
t[k_]:=(Do[AppendTo[tabelle,tabteil[n,k]],
                        {n,0,k}];
       tabelle  = Sort[Flatten[tabelle,1]];
       (tabelleTF=tabelle /. {0->False,
                              1->True});
       Print[MatrixForm[tabelle]];
       MatrixForm[tabelleTF]);
t[3]

Remove[tabteil,tabelle];
tabteil[n_, k_]:= Permutations[
               Table[Ceiling[Floor[n/i]/k],
                              {i,1,k}]];
tabelle = {};
t[k_]:=(Do[AppendTo[tabelle,tabteil[n,k]],
                              {n,0,k}];
       tabelle  = Sort[Flatten[tabelle,1]];
       (tabelleTF=tabelle /. {0->False,
                              1->True});
       Print[MatrixForm[tabelle]];
       MatrixForm[tabelleTF]);
t[4]
```

Und hier der "Output" in der entsprechenden Reihenfolge:

• Et voici l'"Output" dans l'ordre correspondant:

```
??Map

f @@ Transpose[tabelleTF]

MatrixForm[Transpose[
    Join[Transpose[tabelleTF],
    {Table["|",{i,1,
        Length[f @@ Transpose[tabelleTF]]}]},
    {f @@ Transpose[tabelleTF]}]]]

Remove[tabteil,tabelle];
tabteil[n_, k_]:= Permutations[
               Table[Ceiling[Floor[n/i]/k],
                                 {i,1,k}]];
tabelle = {};
t[k_]:=(Do[AppendTo[tabelle,tabteil[n,k]],
                                 {n,0,k}];
       tabelle  = Sort[Flatten[tabelle,1]];
       (tabelleTF=tabelle /. {0->False,
                                 1->True});
       MatrixForm[Transpose[
       Join[Transpose[tabelleTF],
       {Table["|",{i,1,
           Length[f @@ Transpose[tabelleTF]]}]},
       {f @@ Transpose[tabelleTF]}]]]);
(*Anwendung*)
t[5]
```


■ 10.3.5. Verzweigungen
€ Ramification

Probiere aus: • Essaie:

```
?If

?PrintForm

PrintForm[Print"gagag"]

?HorizontalForm

?Write

?WriteString

?SequenceForm

Clear[t];
Do[(t=Table[(If[Random[]>0.5,
    SequenceForm["omumo"],SequenceForm["wvovw"],
    SequenceForm["bubu"]]),{i,7}];
Print[t[[1]],t[[2]],t[[3]],t[[4]],t[[5]],
        t[[6]],t[[7]]]),{k,25}]
```

Verzweigung je nach *Mathematica*-Version:

• Ramification selon la version de *Mathematica*:

```
If[$VersionNumber < 2.0,
    Plot[E^(-x^2) Cos[20x],{x,-2,2},Framed->True],
    Plot[E^(-x^2) Cos[20x],{x,-2,2},Frame ->True]];
```

Mit Switch: • Avec Swith:

```
x=Random[Integer,{1,5}];
Print[x];
Switch[x^2,1,x,4,2x,9,3x,16,4x,25,-5x]

x=Random[Integer,{1,5}];
Print[x];
Switch[x^2,1,x,4,2x,9,3x,16,4x,25,-5x]

x=Random[Integer,{1,5}];
Print[x];
Switch[x^2,1,x,4,2x,9,3x,16,4x,25,-5x]
```

Mit Which: • Avec Which:

```
signum[x_]:=Which[x<0.,-1,x==0.,0,x>0.,1]

Plot[signum[x],{x,-3,3}];
```

■ 10.3.6: Spaghetti-Code (resp. die "Chaos-Erzeugung"...)
€ Code spaghetti (résp. la "génération du chaos"...)

Davon ist abzuraten!
• On le déconseille!

```
?Goto

?Label

?Throw

?Catch
```

■ 10.3.7: Beispiel aus der Statistik
€ Exemple pris de la statistique

Der Median: • La médiane:

```
median[liste_List]:=
    Block[{
            sl,
            len
          },
          len = Length[liste];
          sl = Sort[liste];
          If[
             OddQ[len],
             sl[[(len+1)/2]],
             (sl[[len/2]]+sl[[len/2+1]])/2
          ]
    ]

median[{53,64,78,24,63,78}] // N

median[{76, 56, 23, 78, 34}]

median[{0,1,2,3,4,5,6}]

median[{1,2,3,4,5,6}]  // N

median[{1,2,3,4,5,6,7}]
```

Was ist der Median?
• Qu'est-ce la médiane?

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Uebungen € Exercices

10. Lokale Variablen und prozedurales Programmieren
€ Variables locales et programmation procédurale

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 10 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 10.
WIR94/99

■ Aufgabe 1 € Problème 1

■ Bestimme die Anzahl Iterationsschritte, die notwendig sind, um "Wurzel aus 2" auf 30 Dezimalen mittels der Newton-Methode zu approximieren.
€ Détermine le nombre de pas d'itération qui sont nécessaires pour approximer "racine de 2" jusqu'à 30 décimales par la méthode de Newton.

Starte mit dem Wert 2. Berechne die Wurzel mit der Formel x(i+1)=(x(i)+2/x(i))/2. Erkläre wie diese Formel entsteht. (Löse x^2 - 2 = 0 ...)
• Démarre avec la valeur 2. Calcule la racine par la formule x(i+1)=(x(i)+2/x(i))/2. Explique comment cette formule s'est formée. (Résous x^2 - 2 = 0 ...)

```
Clear[newtonMethode, counter, zahl, approx];
newtonMethode[x_]:=
    Module[{
            counter = 1,
            zahl = N[x,30],
            approx = N[x,40]
        },
        While[approx^2 != zahl,
            Print[counter, ":  ", N[approx,30]];
            approx = (approx + 2/approx)/2;
            ++counter;
        ];
        Print[counter, ":  ", N[approx,30]];
    ]

newtonMethode[2]
```

Vergleich: • Comparaison:

```
N[Sqrt[2],30]
```

■ Aufgabe 2 € Problème 2

■ **Definiere eine Funktion, die als Argumente drei Zahlen aufnimmt und die Summe der Quadrate der beiden grösseren Zahlen ausgibt.**
€ Définis une fonction qui prend 3 nombres comme arguments et sort la somme des carrés des deux nombres plus grands.

Definition: • Définition:

```
Clear[f];
f[x_,y_,z_]:=
    Module[{i,j,liste,min},
        min=Min{x,y,z};
        liste=Sort[{x,y,z}];
        restQuadr=Rest[{x,y,z}]^2;
        Print[Apply[Plus,restQuadr]]
        ]
```

Ausprobieren: • Essayer:

```
{f[0,0,0],f[0,0,1],f[0,1,1],f[1,1,1],f[1,1,2],f[1,2,3]}

f[3]
```

Was ist passiert?
• Qze s'est-il passé?

■ Aufgabe 3 € Problème 3

■ **Annahme: Familien haben Kinder, bis erstmals ein Knabe kommt. Mache eine Simulation mit 1000 Familien um herauszufinden, wieviele Kinder etwa durchschnittlich eine Familie haben wird (Abschätzung). Wieviele Töchter und wieviele Söhne werden in einer Familie zu erwarten sein?**
€ Supposition: Des familles ont des enfants jusqu'à la naissance du 1er garçon. Fais une simulation avec 1000 familles pour obtenir le nombre moyen d'enfants qu'une famille aura (estimation). A combien de filles et à combien de fils pourra s'attendre une famille?

Lösung von Nancy Blachman:
• Solution de Nancy Blachman:


```
Clear[makeFamily];
makeFamily[ ] :=
    Module[{
            children = {}
            },
            While[Random[Integer] == 0,
                AppendTo[children, "girl"]
            ];
            Append[children, "boy"]
        ]

makeFamily::usage="makeFamily[ ] returns a list
                   of children."

Clear[numChildren];
numChildren[n_Integer]:=
    Module[{
            allChildren
            },
            allChildren = Flatten[Table[
                    makeFamily[ ],{n}]];
            {   avgChildren -> Length[
                        allChildren]/n,
                avgBoys -> Count[allChildren,
                            "boy"]/n,
                avgGirls -> Count[allChildren,
                            "girl"]/n
            }
        ]

numChildren::usage="numChildren[n] returns
            statistics on the number of
            children from n families."

numChildren[1000]
```

■ Aufgabe 4 € Problème 4

■ Mischen von Kartenspielen: € Mélanger des jeux de cartes:

Beim perfekten Mischen teilt der Spieler das Kartenspiel genau in der Mitte und mischt das Spiel im "Reissverschlussverfahren": Abwechslungsweise kommt von jeder Seite je eine Karte, wobei sicher sein muss, dass die erste Karte von der ersten Hälfte zuerst "hingelegt" wird (das Spiel wird umgekehrt gehalten). Z.B. wenn man ein Spiel mit 10 Karten hat, ursprönglich geordnet in der Reihenfolge {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, so ist es nach einmal perfektem Mischen in der Reihenfolge {1, 6, 2, 7, 3, 8, 4, 9, 5, 10}.
a) Schreibe eine Funktion "perfektMischen", die das perfekte Mischen einer geraden Anzahl von Karten simuliert.
b) Schreibe die Funktion "wiederOrdnung", die die Anzahl perfekter Mischungen berechnet um das Spiel wieder in die urspüngliche Ordnung zu bringen. Wieviel mal ist es bei 52 Karten?
•
Un joueur qui mélange les cartes à la perfection, divise le jeu de cartes exactement par la moitié et mélange les cartes par la méthode de la "fermeture éclair": De chaque côté il arrive une carte alternativement, mais on doit être sûr que la 1er carte vient de la 1er moitié (on tient le jeu à l'invers). P.ex. si on a un jeu de 10 cartes dans l'ordre d'origin {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, on obtient l'ordre suivant en mélangeant une fois {1, 6, 2, 7, 3, 8, 4, 9, 5, 10}.
a) Ecris une fonction "perfectMischen" qui simule un mélange parfait d'un nombre pair de cartes.
b) Ecris la fonction "wiederOrdnung" qui calcule le nombre de mélange parfaits pour remettre le jeu dans l'ordre d'origine.

```
Clear[mischen, perfektMischen, mischen1,
                perfektMischen1];

mischen::usage="mischen[ ] erzeugt eine Liste
           von perfekt gemischten Zahlen --
           donne une liste de nombres
           mélangés à la perfection.";
mischen[n_]:=
    Module[{matrix1,matrix2},
        listeIn=Range[n];
        matrix1=Partition[listeIn,n/2];
        matrix2=Transpose[matrix1];
        listeOut=Flatten[matrix2];
        Print["Liste_In: ", listeIn, "   ",
         "Liste_Out: ", listeOut]
     ];

perfektMischen::usage="perfektMischen[ ] greift
        auf den Modul mischen[ ] im Falle einer
        eingegebenen geraden natuerlichen Zahl
         -- utilise le Modul mischen[ ] dans
        le cas d-un nombre donné pair,
        naturel.";

perfektMischen[n_Integer]:=
    Module[{listeIn},
        If[(OddQ[n] || n <= 0),
        Print["Bitte gerade, positive Zahl
        eingeben -- S.v.p. entrer nombre
        positif pair."], mischen[n],
        Print["Bitte gerade, positive Zahl
        eingeben -- S.v.p. entrer nombre
```



```
        positif pair."]]
        ];

mischen1::usage="mischen1[ ] erzeugt eine Liste
           von perfekt gemischten Zahlen ohne
           Output -- donne une liste de
           nombres mélangés à la perfection
           sans output.";
mischen1[n_]:=
    Module[{matrix1,matrix2},
        listeIn=Range[n];
        matrix1=Partition[listeIn,n/2];
        matrix2=Transpose[matrix1];
        listeOut=Flatten[matrix2];
        ];

perfektMischen1::usage="perfektMischen1[ ]
        greift auf den Modul mischen[ ]
        im Falle einer eingegebenen geraden
        natuerlichen Zahl. Im Unterschied zu
        perfektMischen[ ]
        wird hier die Ausgabe unterdruekt
        -- utilise le Modul mischen[ ] dans
        le cas d-un nombre donné pair, naturel.
        Par contre à perfektMischen[ ]
        l-output est retenu.";

perfektMischen1[n_Integer]:=
    Module[{},
        If[(OddQ[n] || n <= 0),
        Print["Bitte gerade, positive Zahl
        eingeben -- S.v.p. entrer nombre
        positif pair."], mischen1[n],
        Print["Bitte gerade, positive Zahl
        eingeben -- S.v.p. entrer nombre
        positif pair."]]];

perfektMischen[100]

listeOut
```

```
Clear[mischenAnzahl];

mischenAnzahl::usage="mischenAnzahl[ ] greift
        auf den Modul perfektMischen1[ ] und
        wendet diesen an, bis die ursprüngliche
        Reihenfolge wieder hergestellt ist. Ein
        Zaehler zaehlt die Anzahl Mischablaeufe
        -- utilise le Modul perfektMischen1[ ]
        et l-utilise jusqu-à l-ordre original
        est remis. Un compteur compte le
        nombre de mélanges.";

mischenAnzahl[n_]:=
    Module[{liste, counter=1, matrix1, matrix2},
        mischen1[n];
        liste  =listeIn;
        While[((listeOut != liste) &&
            counter < 10000), Print[counter,
            "  ", listeIn,"  ", listeOut];
            matrix1=Partition[listeOut,n/2];
            matrix2=Transpose[matrix1];
            listeOut=Flatten[matrix2];
            ++counter;
            ]
        Print[counter,"  ",listeIn,"  ",
                                listeOut];
        Print["Anzahl Mischungen --
            Nombre de mélanges:   ", counter];

     ]

perfektMischenAnzahl::usage="
        perfektMischenAnzahl[ ] greift auf den
        Modul mischenAnzahl[ ] im Falle einer
        eingegebenen geraden natuerlichen Zahl  --
        utilise le Modul mischenAnzahl[ ]
        dans le cas d-un nombre donné pair,
        naturel..";


perfektMischenAnzahl[n_Integer]:=
    Module[{},
        If[(OddQ[n] || n <= 0),
        Print["Bitte gerade, positive Zahl
        eingeben-- S.v.p. entrer nombre
        positif pair."], mischenAnzahl[n],
        Print["Bitte gerade, positive Zahl
        eingeben-- S.v.p. entrer nombre
        positif pair."]]]

perfektMischenAnzahl[1]

perfektMischenAnzahl[0]

perfektMischenAnzahl[2]

perfektMischenAnzahl[4]

perfektMischenAnzahl[6]

perfektMischenAnzahl[8]
```



```
perfektMischenAnzahl[10]

perfektMischenAnzahl[12]

perfektMischenAnzahl[14]

perfektMischenAnzahl[16]

perfektMischenAnzahl[18]

(* perfektMischenAnzahl[20] *)

(* perfektMischenAnzahl[52] *)
```

■ Aufgabe 5 € Problème 5

■ Zum Unterschied zwischen "Block" und "Module" € Quant à la différence entre "Block" er "Module"

Betrachte die folgenden Beispiele. Sie sind bis auf die Anweisungen "Block" resp. "Module" identisch. Wieso gibt *Mathematica* verschiedene Resultate aus?

• Considère les exemples suivants. Ils sont identiques, sauf les directives "Block" resp. "Module". Pourquoi est-ce qur *Mathematica* sort des résultats différents?

```
Clear[a, i];
a = i;
i = 3;
Block[{i},
        Table[a, {i, 2}]
    ]

Clear[a, i];
a = i;
i = 3;
Module[{i},
        Table[a, {i, 2}]
    ]
```

■ "Putzmaschine" einsetzen € Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

11. Problemsammlung zur Musterentsprechung (Mustererkennen, musterkonformes Abarbeiten) € Collection de problèmes concernant les correspondances et patrons (reconnaître les patrons, travailler conformement aux patrons)

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 11 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 11.
WIR94/99

■ 11.1. Mustererkennung bei einer Sequenz € Reconnaître un patron lors d'une séquence

■ Ein Beispiel € Un exemple

Bisher bekannte Variablen mit "_": Z.B. in f[x_]:=x wird bei der Anwendung f[y] x durch y ersetzt. Oder bei f[3] x durch 3. Oder bei f[x+2y-Sin[z]] wird x durch x+2y-Sin[z] ersetzt. *Mathematica* erkennt hier das "Muster" des zu ersetzenden Ausdrucks. Wie lässt sich damit z.B. der Mittelwert einer beliebig grossen Zahlenmenge berechnen? Beispiel:

• Variables avec "_" déjà connues: P.ex. dans f[x_]:=x on remplace f[y] x par y à l'application.Ou dans f[3] x par 3. Ou dans f[x+2y-Sin[z]] on remplace x par x+2y-Sin[z]. *Mathematica* reconnaît ici le "patron" de l'expression à remplacer. Comment peut-on calculer p.ex. la valeur moyenne d'un ensemble d'une grandeur quelconque de nombres? Exemple:

```
mittel[a_]:= a;
mittel[a_, b_]:= (a+b)/2;
mittel[a_, b_, c_]:= (a+b+c)/3;  (*etc. ....*)

mittel[4,6]

mittel[9]

mittel[1,2,3]

mittel[1,2,3,4]
```


Was tun? - Wir können auch das Symbol "__" (doppelter Unterstrich) verwenden resp. "___" (dreifach).

• Que faire? Nous pouvons employer aussi le symbole "__" (double sous-ligne) resp. "___" (triple).

```
??__

?___
```

Beispiel: • Exemple:

```
Remove[mittel];
mittel[x__] := Plus[x]/Length[{x}]

mittel[1,2,3,4,5,6,7,8,9]

mittel[a,b,c,d,e,f,g]

mittel[{a,b,c,d,e,f,g}]
```

Um die Funktion auch auf eine Liste anwendbar zu machen, definieren wir:

• Pour rendre la fonction aussi applicable à une liste, nous définissons:

```
mittel[{x__}] := mittel[x]

mittel[{a,b,c,d,e,f,g}]
```

Nun klappt es!

• Voilà, ça marche!

■ 11.2. Nachbau von Funktionen € Copier (imiter) des fonctions

■ 11.2.1. "First" (herauspicken des 1. Elements einer Liste) € "First" (choisir le 1er élément d'une liste)

■ "Von Hand": € "A la main":

Probiere: • Essaie:

```
{a, b, c, d, e, f, g} /. {x_, y___} :> x
(* 3 Unterstriche! *)
```

■ Definition einer Funktion: € Définition d'une fonction:

```
nimmErstes[{x_, y___}] := x
```

Ausprobieren: • Essayer:

```
nimmErstes[{a, b, c, d, e, f, g}]
```

■ 11.2.2. "Last" (herauspicken des letzten Elements einer Liste)
€ "Last" (choisir le dernier élément d'une liste)

■ **"Von Hand":**
€ "A la main":

Probiere: • Essaie:

```
{a, b, c, d, e, f, g} /. {x___, y_} :> y
```

■ **Definition einer Funktion:**
€ Définition d'une fonction:

```
nimmLetztes[{x___, y_}] := y
```

Ausprobieren: • Essayer:

```
nimmLetztes[{a, b, c, d, e, f, g}]
```

■ 11.2.3. "Head"
€ "Head"

■ **"Von Hand":**
€ "A la main":

Probiere: • Essaie:

```
{a, b, c, d, e, f, g} /. h_[x___] :> h
(* 3 Unterstriche! *)
```

■ **Definition einer Funktion:**
€ Définition d'une fonction:

```
nimmKopf[h_[x___]] := h
```

Ausprobieren: • Essayer:

```
nimmKopf[{a, b, c, d, e, f, g}]
```


■ 11.2.4. Neuordnung von Elementen
€ Mettre des éléments dans un nouvel ordre

■ **Zum Beispiel Vertauschung der Reihenfolge**
€ Par exemple renverser l'ordre

Probiere: • Essaie:

```
{{1,3},{2,4},{3,5},{4,6}} /. {x_, y_} :>
                                {y, x}
```

■ 11.3. "Polymorphe" Definitionen
€ Définitions "polymorphe"

■ **Hier geht es um datenabhängige Definitionen!**
€ Il s'agit ici de définitions dépendantes de données!

Gemeint sind da Definitionen, die ihre Form ändern je nach den Daten, auf die sie angewandt werden. Beispiel:
• Il s'agit ici de définitions qui changent leur forme selon les données auxquelles on les applique. Exemple:

```
Remove[pasc];
pasc[n_Integer] := Table[Binomial[n,i],
                                {i,0,n}];
pasc[n_Real   ] := n!;

pasc[5]

pasc[5.5]

Map[pasc,{1,1.2,1.4,1.6,1.8,2,2.2,2.4,2.6,2.8,
                                  3,3.2}]

Plot[n!,{n,0,3}];
```

■ 11.4. Benennung von Ausdrücken
€ Nommer des expressions

■ **Das Muster x:Muster**
€ Le patron x:Exemple

Komplette Ausdrücke können mit Namen versehen werden. Beispiel:
• Les expressions complètes peuvent être pourvues de noms. Exemple:

```
2 + 3 Cos[x] /.
    a:(x_ +y_ Cos[z_]) :>
        {{"a", a},{"x", x},{"y",y},{"z",z}}
```

Identifiziere die Namen!
• Identifier les noms!

■ 11.5. Ausdrücke auffinden, die mit Mustern übereinstimmen
€ Chercher des expressions qui correspondent à des patrons

■ 11.5.1. Beispiele mit "Cases"
€ Exemples avec "Cases"

Probiere aus: • Essaie:

```
??Cases
```

Studieren: • Etudier:

```
Cases[{1, 2.3, {a, b}, x^2 +7, x+y+z},  x_]
```

Einschränkung auf Integers oder Reals:
• Se limiter à des Integers ou Reals:

```
Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                x_Integer]

Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                x_Real]

?|
```

Einschränkung auf eine Alternative von Integers oder Reals:
• Se limiter à une alternative de Integers ou de Reals:

```
Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
       (x_Integer | x_Real)]
```

Einschränkung auf Listen:
• Se limiter à des listes:

```
Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                 x_List]
```

Einschränkung auf "Numbers":
• Se limiter à "Numbers":

```
Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                x_?NumberQ]
```


■ 11.5.2. Beispiele mit "Select"
€ Exemple avec "Select"

Nochmals die letzte Operation, aber mit "Select":
• Encore une fois la dernière opération, mais avec "Select":

```
Select[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                NumberQ]
```

■ 11.5.3. Zusammengesetzte Muster
€ Patrons composés

Z.B. stimmt der Ausdruck "a+b+c+d" überein mit dem Muster "x_ + y__":
• P.ex. l'expression "a+b+c+d" correspond au patron "x_ + y__":

```
Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                  x_+y__]
```

Oder: • Ou:

```
Cases[{1, a + I 5, 3., {a, b}, 2 - 3I, x+y+z},
        x_+I y_]

Cases[{1, a + 5I, 3., {a, b}, 2 - 3I, x+y+z},
        x_+I y_]
```

Wieso hat das jetzt nicht "funktioniert"? - Studiere:
• Pourquoi cela n'a-t-il pas fonctionné? - Etudie:

```
FullForm[a + 5I]

FullForm[a + I 5]
```

("Complex" ist atomar. Man kann also diesen "Head" suchen.)
• ("Complex" est atomique. On peut donc chercher ce "Head".)

```
Cases[{1, a + 5I, 3., {a, b}, 2 - 3I, x+y+z},
        x_Complex]

Cases[{1, a + 5I, 3., {a, b}, 2 - 3I, x+y+z},
       Plus[x_Complex,y_]]
```

■ 11.6. Das Attribut "Orderless"
€ L'attribut "Orderless"

Studiere das folgende Beispiel:
• Etudie l'exemple suivant:

```
f[a_Integer, b_Complex, c_Real] := {a, b, c}
```

```
f[5, 3-8I, 3.346]

f[3-8I, 3.346, 5]
```

Offenbar spielt die Reihenfolge der Argumente eine Rolle. Das kann aber umgangen werden! Dazu muss allerdings für f das Attribut "Orderless" gesetzt und f neu definiert werden! Probiere aus:
• Evidemment l'ordre des arguments joue un rôle. Cependant on peut éviter cela! Pour cela il faut remplacer f par l'attribut "Orderless" et définir à nouveau f ! Essaie:

```
Attributes[f]

ClearAll[f];
SetAttributes[f, Orderless];
f[a_Integer, b_Complex, c_Real] := {a, b, c}

f[3-8I, 3.346, 5]
```

Achtung! Die Attribute Orderless, Flat, HoldAll, FoldFirst, HoldRest und Listable müssen vor der Definition der Funktion gesetzt werden!
• Attention! Les attributs Orderless, Flat, HoldAll, FoldFirst, HoldRest et Listable doivent être placés avant la définition de la fonction!

■ 11.7. Beispiele mit Mustererkennung
€ Exemples avec reconnaissance de patron

■ 11.7.1. Selektives Ausmultiplizieren
€ Multiplications sélectives

■ Beispiel
€ Exemple

Es soll eine Funktion geschrieben werden, die es erlaubt, ausschliesslich nur die Terme auszumultiplizieren, die unter einer Logarithmus-Funktion stehen. Eine Lösung:
• Qu'on écrive une fonction qui permette de multiplier exclusivement les termes qui sont placés sous une fonction de logarithme. Voici une solution:

```
Clear[logAusmult];
logAusmult[x_] := x /. Log[y_] :>
                      Log[Expand[y]]

logAusmult[(4-3x)^5 Log[-(1-x)^3] /(x-1)^3]
```


■ 11.7.2. Manipulation von Klammern
€ Manipulation de parenthèses

■ Beispiele
€ Exemples

Eine Liste von Listen soll in eine Liste verwandelt werden:
• Qu'on transforme une liste de listes en une liste:

```
{{a, b, c, d},{e, f, g}} /.
                {{x__},{y__}} :> {x, y}

{{a, b, c, d},{e, f, g}} /.
                {x__} :> x

Remove[f];
Map[f,{{a, b, c, d},{e, f, g}}]

Map[f,{{a, b, c, d},{e, f, g}}] /.
                f[{x__}] :> f[x]
```

■ 11.7.3. Eine eigene Definition von "Map"
€ Une propre Définition de "Map"

Studiere das folgend Beispiel:
• Etudie l'exemple suivant:

```
fktMap[f_, {}] := {};
fktMap[f_, {a_, b___}] :=
           Prepend[fktMap[f, {b}], f[a]];


fktMap[Sin,{a, 2, 3, 4, 5, 6, 7}]
```

■ 11.7.4. Ziffern zählen
€ Compter des Chiffres

■ Beispiel
€ Exemple

Es soll die Frage beantwortet werden, ob es sechs aufeinanderfolgende Ziffern "9" gibt unter den ersten 1000 Dezimalstellen von Pi. Falls ja, so soll die Anzahl Dezimalstellen zwischen Komma und dieser Sechsergruppe ausgegeben werden. Studiere die Lösung:
• Il faut répondre à la question si le chiffre "9" apparaît 6 fois de suite parmi les 1000 premières unités décimales de Pi. Si cela est le cas, il faut sortir le nombre d'unités décimales situées entre la virgule et ce groupe de six "9". Etudie la situation:

```
Characters[ToString[1234567]]

charListe = Characters[ToString[N[Pi, 10]]]
```

Ohne Output (1000....)!
Sans output (1000...)!

```
charListe = Characters[ToString[N[Pi, 1000]]];

findeNeun[{"3", ".", x___,
        "9","9","9","9","9","9",___}] :=
                Length[{x}];

findeNeun[charListe]
```

■ 11.7.5. Uebergabe von Optionen (Options)
€ Transmission d'options

■ Beispiel
€ Exemple

Studiere: • Etudie

```
Needs["Graphics`Graphics`"]

??Graphics
```

Beachte: • Observe:

Der zu verwendende Modul sieht etwa so aus
(- man sieht sofort, was übergeben werden kann...):
• Le module à utiliser a environ cet aspect
(- on voit tout de suite ce qui peut être transmis...):

```
LogLogPlot[f_, {x_, xmin_, xmax_}, opts___] :=
      Module[{r,
        g = ParametricPlot[{Log[10,x], Log[10,f]}, {x, xmin, xmax},
            Ticks->{LogScale, LogScale},
            FrameTicks -> {LogScale, LogScale,
                  LogScale, LogScale},
            DisplayFunction -> Identity, opts]},
    r = PlotRange[g];
    Show[g, DisplayFunction -> $DisplayFunction,
            PlotRange -> r, AxesOrigin -> Map[#[[1]]&,r]]]

    LogLogPlot[x^2, {x, 0.2, 12},
            PlotLabel -> "LogLogPlot von x^2",
            Frame -> True];
```

Versuche dieses Beispiel mit anderen Optionen!
• Essaie cet exemple avec d'autres options!

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

11. Problemsammlung zur Musterentsprechung (Mustererkennen, musterkonformes Abarbeiten)
€ Collection de problèmes concernant les correspondances et patrons (reconnaître les patrons, travailler conformement aux patrons)

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 11 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 11.
WIR94/99

■ Aufgabe 1 € Problème 1

■ Schreibe eine Funktion "geometrisches Mittel", die das geometrische Mittel einer Liste von Zahlen ausgibt.
Ecris une fonction "moyenne géométrique", qui donne la moyenne géométrique d'une liste de nombres.

Z.B. das geometrische Mittel von {a,b,c,d,e} ist (a b c d e)^(1/5).
• P.ex. la moyenne géométrique de {a,b,c,d,e} est (a b c d e)^(1/5).

```
Clear[geometrMittel];
geometrMittel[{x__}]:= Times[x]^(1/Length[{x}])

geometrMittel[{1,1,1,1,1}]

geometrMittel[{2,16,8,4,32}]

geometrMittel[{2,3,5}]
```

```
geometrMittel[{2,3,5}] // N
```

■ Aufgabe 2 € Problème 2

■ Schreibe eine eigene Version der "Join-Funktion". € Ecris ta propre version de la fonction "Join".

"Join" fuegt zwei Listen hintereinander im Gegensatz zur Mengenvereinigung, wo mehrfach vorkommende Elemente nur einmal aufgefuehrt werden.
• "Join" rassemble deux listes l'une après l'autre. Dans la réunion des ensembles par contre les éléments doubles, triples etc. n'apparaissent qu'une seule fois.

```
Clear[joinFkt];
joinFkt[{x__},{}]:= {x};
joinFkt[{},{x__}]:= {x};
joinFkt[{x__},{y__}]:= Flatten[{{x},{y}}]
```

Ausprobieren: • Essayer:

```
joinFkt[{a,b,c,d},{e,f,g}]

joinFkt[{a,b,c,d},{a,e,b,f,g}]

joinFkt[{a,b,c,d},{}]

joinFkt[{},{}]
```

Was ist passiert? Erweitere die Definition sinngemäss!
• Que s'est-il passé? Elargis la définition d'une façon sensée!

■ Aufgabe 3 € Problème 3

■ Baue eine Funktion "Fold" wie unten beschrieben € Construis une fonction "Fold" comme il est décris ci-dessous.

a) Implementiere wie nachstehend folgt. Beantworte dann die nachstehenden Fragen b), c), d) und vergleiche mit den implementierten Funktionen "Fold" resp. "FoldList".
• "Implement" comme il suit. Réponds aux questions suivantes b), c), d) et compare avec les fonctions "implémentées" "Fold" resp. "FoldList".

```
Clear[falte];
falte::usage = "falte[f, basis, liste] ergibt
           --donne  f[...[f[f[basis,x1]x2],...xn]
           wobei --étant  list = {x1,x2,...,xn}";
falte[f_,basis_,{}]:=basis;
falte[f_,basis_,{x1_,xrest___}]:=
       falte[f,f[basis,x1],{xrest}];
```


b) Was macht "falte[Plus,0,liste]"? Z.B. liste={a,b,c}.
• Que fait "falte[Plus,0,liste]"? P.ex. liste={a,b,c}.

```
falte[Plus,0,{a,b,c}]

falte[Plus,0,{1,2,3,4}]

falte[Plus,5,{1,2,3,4}]
```

c) Was macht "falte[Max,-Infinity,liste]"? Z.B. liste={a,b,c}.
• Que fait "falte[Max,-Infinity,liste]"? P.ex. liste={a,b,c}.

```
falte[Max,-Infinity,{a,b,c}]

falte[Max,-Infinity,{6,8,3}]

falte[Max,9,{6,8,3}]
```

d) Vergleiche mit "Fold" und "FoldList":
• Compare avec "Fold" et "FoldList":

```
??Fold

??FoldList
```

■ Aufgabe 4 € Problème 4

■ Schreibe die APL-Funktion "Deal" resp. "?" in *Mathematica*: € Ecris la fonction APL "Deal" resp. "?" dans *Mathematica*:

"L ? R" in APL wählt (streicht) aus dem "Range[r]" pseudo-zufällig L Zahlen aus, ohne sie in R zu ersetzen.
• "L ? R" dans APL choisit (biffe) dans le "Range[r]" d'une façon pseudo-aléatoire L nombres, sans les remplacer dans R.

a) Was ist "Range" - Wie erzeuge ich die "Zufallszahlen"?
• Qu'est "Range" - Comment générér les "nombres aléatoires"?

```
??Range

Range[14]

??Random
```

b) Die Beschreibung der Funktion "deal" in *Mathematica*:
• La description de la fonction "deal" dans *Mathematica*:

```
RemoveAll[deal];
deal::usage="deal[n,r] streicht zufällig n
       Integers aus der Liste  Range[r] ohne
       Ersetzung.
       -- biffe de façon aléatoire n Integers
       dans la liste Range[r] sans les
       remplacer."
```

c) Hier eine mögliche Variante:
• Voici une variante possible:

```
Clear[deal];
deal[0,r_] :=Range[r];
deal[n_,r_]:=Sort[Rest[RotateLeft[deal[n-1,r],
                      Random[Integer,{1,r}]]]]
```

d) Ausprobieren:
• Essaier

```
deal[2,6]
```

e) Erklärungen:
• Explications:

```
??Rest

??RotateLeft

RotateLeft[{a,b,c},11]

??Sort
```

■ Aufgabe 5 € Problème 5

■ Zu Funktionen, die als Argument eine variable Zahl von Listen aufnehmen: Quant aux fonctions qui acceptent comme argument un nombre variable de listes:

Das Muster "x___List" als Argument in einer Funktion lässt eine variable Zahl von Listen zu. Wie kann man auf einen Schlag alle Längen einer Anzahl beteiligter Listen finden? Schreibe eine Funktion dafür!
• Le patron "x___List" comme argument dans une fonction permet un nombre variable de listes. Comment peut-on, d'un coup, trouver touttes les longeurs d'un nombre de listes faisant partie du problème? Ecris une fonction pour cela!

```
Clear[listenLaengen];
listenLaengen[x___List]:=Map[Length, {x}]
```

Ausprobieren: • Essayer:

```
listenLaengen[{1,2,3}, {1,2,3,4,5}, {1,1,1,1},
              {a,b,c,d,e,f,g}]
```


■ Aufgabe 6 € Problème 6

■ Lauflängencodierung in Listen (Aäufigkeitszählingen) € Conter des fréquences dans les listes

Konstruiere eine kurze *Mathematica*-Funktion, die die Häufigkeit der Elemente einer Liste zählt. Die Funktion "laufCod" soll eine Liste von Paaren {Listenelement, absolute Häufigkeit} ausgeben Beispiel:
liste = {e,b,a,b,e,c,c,d,e,a,a,d,u};
laufCod[liste] soll das Resultat {{a, 3}, {b, 2}, {c, 2}, {d, 2}, {e, 3}, {u, 1}} ergeben. Die Funktion "laufDeCod" soll aus dem Output von "laufCod" wieder die ursprünglich Liste machen.
• Construis une fonction de *Mathematica* courte qui compte la fréquence des éléments d'une liste. La fonction "laufCod" doit sortir une liste de paires {élément de liste, fréquence absolue}. Exemple:
liste = {e,b,a,b,e,c,c,d,e,a,a,d,u};
laufCod[liste]doit donner le résultat
{{a, 3}, {b, 2}, {c, 2}, {d, 2}, {e, 3}, {u, 1}}. La fonction "laufDeCod" doit transformer l'output de "laufCod" de nouveau dans la liste originale.

■ a) laufCod

```
laufCod[liste_List]:=Transpose[{Union[liste],
       Table[Count[liste,Part[Union[
       liste],n]], {n,Length[Union[liste]]}]}]
```

(Hat Schweiss gekostet!) Aber jetzt ausprobieren:
• (Cela nous a fait transpirer!) Mais maintenant on essaie:

```
RemoveAll[liste];
liste = {1,1,2,2,2,3,3,3,3};
output = laufCod[liste]

liste = {e,b,a,b,e,c,c,d,e,a,a,d,u};
output = laufCod[liste]
```

■ a) laufDeCod

```
laufDeCod[output_]:= If[MatrixQ[output],
            output/.{x_,n_}:> Flatten[
                        Table[x,{n}]],                    Print["Bitte
          Print["Bitte Matrix eingeben --
                 S.v.p. entrer la matrice"]]

laufDeCod[{{1,2},{3,5},{4,3}}]

laufDeCod[{{a,3},{b,5},{c,2},{d,4}}]
```

■ **c) Eine andere Variante für "laufCod" mit Ersetzungsregeln**
(funktioniert nur för eine Liste von Integers)
€ Une autre variante pour "laufCod" avec des règles de remplacement
(ne fonctionne que pour une liste d'Integers)

```
RemoveAll[f];
f[liste_List]:=Module[{i},
i=0;
a=liste;
a//.{y___,x_,x_,z___}:>
{y,x,++i b,x,z}//.{y___,x_Integer,w_Integer,
                   z___}:>
{y,x,d, c,w,z}/. {x___}:>{c, x, d}//.
{x___,y___ b,z___}:>{x,z}//. {x___,b,
                  z___}:>{x,z}//.
{y___,c,x__,d,z___}:> {y,{x},z}
];
laufCod1[liste_List]:={Map[First,f[liste]],
                       Map[Length,f[liste]]}

f[{1,1,1,2,2,2,2,2,3,3,3,3,4,5,5}]

laufCod1[{1,1,1,2,2,2,2,2,3,3,3,3,4,5,5}]

laufCod1[{0,0,0,1,1,2,2,2,3,3,3,3,4,4,4,5,5,6,
                  6,6}]
```

Was ist los?
• Que se passe-t-il?

```
laufCod1[{a,a,a,1,b,b,b,2,2,2,3,3,3,3,4,4,4,5}]
```

d) Noch eine andere Variante:
• Encore une autre variante:

■ **d) Noch eine andere Variante für "laufCod" mit Hilfe anonymer**
Funktionen (vgl. Kap. 12):
€ Encore une autre variante pour "laufCod" à l'aide de fonctions anonymes (comparer chap. 12):

```
Attributes[mal]={Flat};
mal[{elem_,m_},{elem_,n_}]:=mal[{elem,m+n}];
laufCod2[liste_List]:= List @@ mal @@({#,1}& /@
                       liste)

laufCod2[{1,1,1,2,2,2,2,2,3,3,3,3,4,5,5}]
```

Wie arbeitet dieses Programm?
• Comment ce programme travaille-t-il?

steht für "anonyme Variable", & für "anonyme Funktion", /@ für "Map".
Durchlaufen wird die Liste.
• # signifie "variable anonyme", & signifie "fonction anonyme", /@ signifie "Map". La liste est parcourrue.

```
out1=({#,1}& /@ {1,1,1,2,2,2,2,2,3,3,3,3,4,5,5})
```


@@ steht für "Apply".
• @@ signifie "Apply".

```
out2=mal @@out1

??List

out3=List @@ out2
```

Probiere eigene Varianten aus!
• Essaie des variantes à toi!

■ Testgelände:
€ Terrain de test:

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

12. Anonyme Funktionen
€ Fonctions anonymes

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 12 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach.... Indication:
Lire le chapitre 12.
WIR94/99

Anonyme Funktionen sind solche, die keinen Namen haben ....
Hier werden wir sehen, was es damit um sich hat!
• Les fonctions anonymes sont celles qui n'ont pas de nom...
Ici nous verrons ce qui en est d'elles!

■ 12.1. "Function"
€ "Fonction"

■ **"Function" ist eine allgemeine Funktion ohne speziellen Namen, die gerade "unmittelbar" angewandt wird.**
€ "Function" est une fonction générale sans nom spécial, on peut l'utiliser "immédiatement".

Studiere: • Etudie:

```
??Function
```

Beispiele: • Exemple:

```
Function[x, x^2][4]
```

```
Function[x, x^2 - x][5]
```

```
Function[x, x^2][{1, 2, 3, 4, 5, 6, 7}]
```

Einfach, nicht? • Simple, n'est-ce pas?

■ 12.2. Anwendung auf Datenselektion
€ Application à la sélection de données

■ 12.2.1. Mit Hilfe einer klassischen Funktion mit Name (hier "groes45")
€ A l'aide d'une fonction classique avec nom (ici "groes45")

Beispiel (erst Daten erzeugen, dann Funktion definieren, dann anwenden):
• Exemple (d'abord générer des données, ensuite définir la fonction, puis appliquer):

```
data = Table[Random[Integer, {0, 100}], {14}]
```

```
groes45[x_] := x > 45
```

```
Map[groes45,data]
```

```
groes45[data]
```

Anwendung zur Selektion:
• Application à la séléction:

```
Select[data, groes45]
```


■ 12.2.2. Mit Hilfe einer anonymen Funktion
€ A l'aide d'une fonction anonyme

Beispiel (erst ausprobieren):
• Exemple (essayer d'abord):

```
Function[x, x > 45][68]
```

```
Function[x, x > 45][24]
```

Anwendung zur Selektion:
• Application à la sélection:

```
Select[data, Function[x, x > 45]]
```

■ 12.2.3. Mit Hilfe der Abkürzung "#" für die Variable der anonymen Funktion
€ A l'aide de l'abréviation "#" pour les variables de la fonction anonyme

```
Select[data, Function[# > 45]]
```

■ 12.2.4. Mit Hilfe der Abkürzung "( )&" für Function[ ]
€ A l'aide de l'abréviation "( )&" pour Function[ ]

```
Select[data,  (# > 45)& ]
```

■ 12.3. Neuauflage einer früher definierten Funktion
€ Nouvel emploi d'une fonction définie auparavant

■ **Präzision 20 Stellen bei numerischen Rechnungen**
€ Précision de 20 unités lors de calculs numériques

Das Beispiel:
• L'exemple:

```
nZwanzig[x_] := N[x, 20]
```

```
$Post = nZwanzig
```

```
Sin[1]
```

Mit anonymer Funktion:
• Avec une fonction anonyme:

```
$Post := N[#, 40]&     (* vierzig *)

Sin[1]
```

■ 12.4. Transformation von Wertepaaren in Regeln
€ Transformation de paires de valeurs en règles

■ **Aus zwei Listen {x1, x2, x3,... } und {y1, y2, y3,... } soll eine Liste {x1 - > y1, x2 -> y2, x3 -> y3,... } gemacht werden.**
€ Faire de deux listes {x1, x2, x3,... } et {y1, y2, y3,... } une seule {x1 - > y1, x2 -> y2, x3 -> y3,... }.

Listen erzeugen:
• Générer des listes de paires:

```
list1 = {x1, x2, x3, x4, x5, x6};
list2 = {y1, y2, y3, y4, y5, y6};
```

Paarliste erzeugen:
Générer des listes de paires:

```
Transpose[{list1, list2}]
```

Anonyme Funktion (#[[1]] -> #[[2]])& auf die Liste setzen (mit "Map"):
• Appliquer des fonctions anonymes (#[[1]] -> #[[2]])& à des listes (avec "Map"):

```
Map[(#[[1]] -> #[[2]])&, Transpose[{list1,
                          list2}]]
```

■ 12.5. Mehrere Argumente
€ Plusieurs arguments

Beispiel: • Exemple:

```
Function[{x, y}, x > y][1, 2]

Function[{x, y}, x > y][2, 1]

Function[{x, y}, x > y][u, 2]
```

Was passiert, wenn die Anzahl der übergebenen Argumente nicht stimmt?
• Que se passe-t-il quand le nombre des arguments transmis n'est pas correct?

```
Function[{x, y}, x > y][1]
```



```
Function[{x, y}, x > y][1, 2, 3]

Function[{x, y}, x > y][a, b, c]
```

Bei der Kurzschreibweise mit "&" und "#" kommt nun das Problem der Identifikation der verschiedenen Variablen. Die Variablen können dabei nummeriert werden. Beispiel:
• Les abréviations "&" et "#" posent le problème de l'identification des différentes variables. On peut numéroter les variables. Exemple

```
??#

?##

??&

?&&

(#1 < #2)&[4, 6]
```

Die anonyme Funktion ist jetzt (#1 < #2)&.
• La fonction anonyme est maintenant (#1 < #2)&.

■ 12.6. Daten filtern
€ Filtrer des données

■ **Aufgabe**
€ Problème

Ersetze eine Datenmenge nach einem "Filter" durch die Menge der gewichteten Mittelwerte je dreier sich folgender Elemente. Studiere dazu das folgende Beispiel!
• Remplace un ensemble de données après un "Filtre" par l'ensemble des moyennes pondérées de trois éléments qui se suivent. Etudie l'exemple suivant!

■ **Lösung in Einzelschritten:**
€ Solution par étapes:

Zuerst Filter und Daten generieren:
• Générer d'abord le filtre et les données:

```
filter = {0.1, 0.8, 0.1};
datenF  = Table[Random[Integer, {0, 100}]/10,
                                  {10}]
```

Ordnen in Dreiergruppen, die dann gewichtet gemittelt werden können:
• Ordonner en groupes de trois dont on peut faire ensuite la moyenne pondérée:

```
matrix = Partition[datenF, Length[filter], 1]

% // MatrixForm
```

Die Gewichtung geschieht, indem man die erste Zahl einer Dreiergruppe mit der ersten Zahl des Filters multipliziert, die zweite Zahl des Filters mit der zweiten der Dreiergruppe etc. und das jeweils addiert. Diese Multiplikation mit nachfolgender Addition ist jeweils ein Skalarprodukt entsprechender Vektoren:
• L poneération se fait en multipliant le 1er nombre d'un group de trois avec le 1er nombre du filtre, le 2e nombre du filtre avec le 2e nombre du groupe etc.. On en fait chaque fois l'addition. Cette multiplication suivie de l'addition scalaire des vecteurs correspondants:

```
Map[filter .#&, matrix]
```

▪ **Lösung alles auf einmal:**
€ Solution en un coup:

```
zusrollen[filterVar_List,
    dataVar_List]:= Map[filterVar . #&,
    Partition[dataVar, Length[filterVar], 1] ]

zusrollen[filter, datenF]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Uebungen € Exercices

12. Anonyme Funktionen
€ Fonctions anonymes

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 12 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 12.
WIR94/99

■ Aufgabe 1 € Problème 1

▪ **Leseübung: Schreibe benannte Funktionen, die dasselbe ausgeben wie die folgenden anonymen Funktionen:**
€xercice de lecture: Ecris des fonctions dénommées qui sortent la nême chose que les fonctions anonymes suivantes:

(a) Anonym: (#^3)&
• Anonyme: (#^3)&

```
{(#^3)&[0],(#^3)&[1],(#^3)&[2],(#^3)&[3],(#^3)&[4]}
```

Benannt: • Dénommé:

```
f[x_]:=x^3;
{f[0],f[1],f[2],f[3],f[4]}
```

(b) Anonym: (#^#)&
• Anonyme: (#^#)&

```
{(#^#)&[0],(#^#)&[1],(#^#)&[2],(#^#)&[3],(#^#)&[4],
(#^#)&[5]}
```

Benannt: • Dénommé:

```
Remove[f];
f[x_]:=x^x;
{f[0],f[1],f[2],f[3],f[4],f[5]}
```

Benannt: • Dénommé:

```
f[x_]:=x^3;
{f[0],f[1],f[2],f[3],f[4]}
```

(c) Anonym: {#,#^2}&
• Anonyme: (#^#^2)&

```
{{#,#^2}&[0],{#,#^2}&[1],{#,#^2}&[2],{#,#^2}&[3],
{#,#^2}&[4],{#,#^2}&[5]}
```

Benannt: • Dénommé:

```
Remove[f];
f[x_]:={x,x^2};
{f[0],f[1],f[2],f[3],f[4],f[5]}
```

(d) Anonym: If[#>0,#,-#]&
• Anonyme: If[#>0,#,-#]&

```
{If[#>0,#,-#]&[-2],If[#>0,#,-#]&[-1],If[#>0,#,
-#]&[0], If[#>0,#,-#]&[1],If[#>0,#,-#]&[2]}
```

Benannt: • Dénommé:

```
Remove[f];
f[x_]:=Abs[x];
{f[-2],f[-1],f[0],f[1],f[2],f[3]}
```

(e) Anonym: (#/.x->y)&
• Anonyme: (#/.x->y)&

```
{(#/.x->y)&[-2],(#/.x->y)&[-1],(#/.x->y)&[0],
(#/.x->y)&[1],(#/.x->y)&[2],(#/.x->y)&[3],
(#/.x->y)&[x],(#/.x->y)&[y],(#/.x->y)&[z]}
```

Benannt: • Dénommé:

```
Remove[f];
f[z_]:=z /. x->y;
{f[-2],f[-1],f[0],f[1],f[2],f[3],f[x],f[y],f[z]}
```

■ Aufgabe 2 € Problème 2

■ **Schreibe eine anonyme Funktion, die die 3. Potenz des Arguments berechnet.**
€ Ecris une fonction anonyme qui calcule la puissance 3 des arguments.

Zum Beispiel (#^3)&
• Par exemple (#^3)&

```
{(#^3)&[-3],(#^3)&[-2],(#^3)&[-1],(#^3)&[0],
 (#^3)&[1],(#^3)&[2],(#^3)&[3]}
```


■ Aufgabe 3 € Problème 3

■ **Verwende "Select" sowie eine anonyme Funktion, um aus einer Liste von Paaren, diejenigen Paare herauszusuchen, bei denen die erste Zahl grösser ist als die zweite.**
€ Utilise "Select" ainsi qu'une fonction anonyme pour choisir dans une liste de paires ces paires dont le premier nombre est plus grand que le deuxième.

Ein Beispiel:
• Un exemple:

```
newList = {{1,2},{2,1},{3,1},{2,3},{6,4},{4,7}};
Select[newList,(#[[1]] > #[[2]])&]
```

■ Aufgabe 4 € Problème 4

■ **Definiere eine anonyme Funktion, um die Option "PlotLabel" bei den plotgenerierenden Kommandos zu setzen. Bilde die Funktion ab auf ContourPlot, DensityPlot, ParametricPlot, Plot, Plot3D.**
€ Définis une fonction anonyme pour joindre l'option "PlotLabel" aux commandements qui génèrent des plots. "Applique" la fonction à ContourPlot, DensityPlot, ParametricPlot, Plot, Plot3D.

Was ist "PlotLabel"?
• Qu'est "PlotLabel"?

```
??PlotLabel
```

Wie wird es eingesetzt?
• Comment l'applique-t-on?

```
??Plot
```

Ein Plot: • Un plot:

```
Plot[Sin[x],{x,-5,5}];
```

Mit Label: • Avec Label:

```
Plot[Sin[x],{x,-5,5},PlotLabel->FontForm[
"1. Sinus,\n zwischen {-5,5}",
                    {"Courier-Bold",15}]];
```

Mit Hilfe einer anonymen Funktion:
• A l'aide d'une fonction anonyme:

```
(Plot[Sin[x],{x,-5,5},PlotLabel->#])&[FontForm[
"2. Sinus,\n zwischen {-5,5}",
                  {"Courier-Bold",15}]];

(Plot[Sin[x],{x,-5,5},PlotLabel->#])&[FontForm[
"Sinus,\n zwischen {-5,5}",{"Courier-Bold",15}]];
```

Etwas geschraubter:

• Un peu plus sophistiqué:

```
Apply[Plot,{#[[1]],#[[2]],#[[3]]}]&[{Sin[x],
  {x,-5,5},(PlotLabel->#)}&[FontForm["3.
  Sinus,\n zwischen {-5,5}",{"Courier-Bold",15}]]];
```

Noch etwas geschraubter:

• Ecore plus sophistiqué:

```
Plot @@ {#[[1]],#[[2]],#[[3]]}&[{Sin[x],{x,-5,5},
 (PlotLabel->#)}&[FontForm["4.
 Sinus,\n zwischen {-5,5}",{"Courier-Bold",15}]]];
```

Ein Versuch, die Options zu ändern:

• Un essai de changer les options:

```
Options[Plot]
```

```
Unprotect[Options[Plot]];
Drop[Options[Plot],{11}];
Options[Plot]=Union[Options[Plot],{PlotLabel->
FontForm["5. Neues Bild",{"Courier-Bold",15}],
Frame->True}];
(*Protect[Plot]*)
```

```
Attributes[Plot]
```

```
{HoldAll}
```

```
Options[Plot]
```

```
Unprotect[Plot];
Drop[Options[Plot],{11}];
Options[Plot]=Union[Options[Plot],{PlotLabel->
FontForm["5. Neues Bild",{"Courier-Bold",15}],
Frame->True}];
(*Protect[Plot]*)
```

```
Attributes[Plot]
```

```
Options[Plot]
```

Was hat er nicht getan?

• Que n'a-t-il pas fait?

```
Plot[Sin[x],{x,-5,5}];
```

Wieso kam der Titel nicht?

• Pourquoi le titre n'est-il pas apparu?

```
Show[%];
```


Uebungsfeld: • Terrain d'exercice:

■ Aufgabe 5 € Problème 5

■ Schreibe nochmals eine Funktion, die die Häufigkeit eines Elements in einer Liste zählt. € Ecris encore une fonction qui compte la fréquence d'un élément dans une liste.

Verwende Union, Count und Map.

• Utilise Union, Count et Map.

Diese Funktion zählt die Häufigkeit von verschiedenen Elementen
einer Liste:

• Cette fonction compte la fréquence de différents éléments d'une liste:

```
frequenz[x_List]:= Module[{elemente=Union[x]},
                Map[{#,Count[x,#]}&,elemente]];
frequenz::usage="frequenz[x_List] gibt eine Liste \
           von verschiedenen Elementen zusammen
           mit ihrer Häufigkeit ---- \
           donne une liste avec différents \
           éléments avec leurs fréquences.";
testListe = Table[Random[Integer,{1,20}],{18}];
Sort[testListe]
```

```
frequenz[testListe]
```

```
??frequenz
```

Probiere eigene Listen!

• Essaie des listes à toi!

■ Aufgabe 6 € Problème 6

■ Graphisches € Graphiques

Eine Aufgabe in mehreren Schritten

• Un problème en plusieurs étapes

■ **(a) Generiere eine Liste von 10 Pseudo-Zufallspunkten im Raum!**
€ Génère une liste de 10 points pseudo-aléatoires dans l'espace!

```
punkte = Partition[
         Table[Random[Integer,{1,20}],{30}],3]
```

■ **(b) Generiere eine Liste von 10 Pseudo-Zufallszahlen für das Grau-Niveau!**
€ Génère une liste de 10 points pseudo-aléatoires pour le niveau gris!

```
grau = Table[Random[Real,{0,1}],{10}]
```

■ **(c)**

```
grau1=Partition[grau,1]

trans=Transpose[{punkte,grau1}];trans

RemoveAll[mix];
mix=Map[Join[#[[1]],#[[2]]]&,trans]

menge=Map[({PointSize[0.03],GrayLevel[#[[4]]],
           Point[{#[[1]],#[[2]],#[[3]]}]
           })&,
           mix];
menge

Show[Graphics3D[menge]];
```

Einige Erklärungen:
• Quelques explications:

```
??Show

??Graphics3D

??DefaultColor

??GrayLevel

{PlotStyle->GrayLevel[1]};
Show[Graphics3D[{PointSize[0.5],GrayLevel[0.5],
Point[{1,1,1}]
}]];

??#
```


■ Aufgabe 7 € Problème 7

■ Das Problem deines "signifikanten Partners" € Le problème de ton "partenaire significatif"

Hier geht es um die Analyse einer besondern Strategie zum Auffinden Deines "signifikanten Partners". Angenommen, da seinen n Personen des andern Geschlechts, die Du in Betracht ziehen willst. Du selbst bist noch ledig, jedoch heiratswillig. Jeder der n Personen ist ein unterscheidbarer Wert zugewiesen, den Du nicht kennst, bis Du das Individuum triffst. Jede Person kannst Du nur einmal treffen. Nach jedem Treffen musst Du entscheiden, ob das eben getroffene Individuum Dein "signifikanter Partner" ist und ihm dann einen Heiratsantrag stellen. Im Falle der Ablehnung gehst Du zm nächsten Treffen, wo dieselben Regeln gelten. Eine abgelehnte Person kannst Du nicht nochmals treffen, entsprechend den hier geltenden Gepflogenheiten.
Du sollst jetzt folgende Strategie analysieren: Du triffst (n/e) Partner als Testmenge, e = Eulersche Zahl. Nachdem Du jetzt einen Eindruck gewonnen hast, wählst Du den ersten Partner Deines Wunsches, der Dir besser gefällt als alle bisherigen Partner.
Wähle n = 30 und lasse eine Simulation 50 mal laufen. Bezeichne im voraus einen besten Partner. Ein Lauf sei positiv, wenn Du den vorher bezeichneten besten Partner durch das Verfahren auch findest.
•
Il s'agit ici d'une stratégie spéciale pour trouver ton "partenaire signicatif". Supposons n personnes de l'autre sexe que tu veux considérer. Toi-même tu es encore célibataire, mais décidé de te marier. A chacun des n personnes est attribuée une valeur qu'on peut distinguer, que tu ne connais pas, jusqu'à ce que tu rencontres l'individu. Tu ne peux rencontrer chaque personne qu'une seule fois. après chaque rencontre tu dois décider si l'individu que tu viens de rencontrer est ton "partenaire significatif" et lui faire ensuite une proposition de mariage. En cas de refus tu te rends au prochain rendez-vous, où valent les mêmes règles. Une personne refusée tu ne peux plus la rencontrer, selon les us et coutumes en vigeur ici.
Analyse la stratégie suivante: Tu rencontres (n/e) partenaires comme ensemble de test, e = nombre d'Euler. Après cette première impression tu choisis le partenaire de les autres. Choisis n = 30 et fais marcher la simulation 50 fois. Désigne à l'avance le meillelur partenaire. Un parcours soit positif si tu trouves le partenaire désigné à l'anvance par cette méthode.

■ **(a) Berechne den prozentualen Anteil der positiven Läufe.**
€ Calcule la partie proportionelle des parcours positifs:

Definiere zuerst eine Funktion "zufall", die eine Liste von Leuten in zufälliger Reihenfolge wieder ausgibt.
• Définis d'abord une fonction "hasard", qui sort une liste de personnes en ordre aléatoire.

Definiere zuerst die Funktion "zufall1". Diese Funktion stellt die Liste rekursiv zufällig um.
• Définis d'abord la fonction "hasard". Cette fonction rédonne la liste de Façon récursive aléatoire.

```
zufall1[{}]:={};
zufall1[personen_List]:=
  Block[{n=Random[Integer,{1,Length[personen]}]},
    Prepend[zufall1[Drop[personen,{n}]],
    personen[[n]]]];
```

Einige Details zum Test:
• Quelques détails quant au test:

```
personen={a,b,c,d};
zufall1[personen]

n=Random[Integer,{1,Length[personen]}]

n=Random[Integer,{1,Length[personen]}];
Drop[personen,{n}]

zufall1[Drop[{a,b,c,d},{2}]]
```

Oder definiere dann die Funktion "zufall2". Diese Funktion tut dasselbe wie "zufall1":

• Ou bien définis la fonction "zufall2". Cette fonction fait la même chose que "zufall2":

```
zufall2[{}]:={};
zufall2[personen_List]:=
 Block[{n=Random[Integer,{0,Length[personen]-1}],
                shifted},
        shifted=RotateLeft[personen,n];
        Prepend[zufall2[Drop[shifted,1]],
        shifted[[1]]]];
```

Einige Details zum Test:

• Quelques détails quant au test:

```
personen={a,b,c,d};
zufall2[personen]

n=Random[Integer,{0,Length[personen]-1}]

n=Random[Integer,{0,Length[personen]-1}];
shifted=RotateLeft[personen,n]

n=Random[Integer,{0,Length[personen]-1}];
shifted=RotateLeft[personen,n];
Drop[shifted,1]
```

Oder definiere dann die Funktion "zufall" mit Hilfe der Funktion "Sort".
Hier werden Paare {a,b} benützt, wo bei a eine Zufallszahl und b die
Ordnungszahl der Person ist. Die Paare werden dann nach a geordnet. Diese Funktion tut dasselbe wie "zufall1":

• Ou bien définis la fonction "zufall" à l'aide de la fonction "Sort". Ici on emploie des paires {a,b}, a étant un nombre aléatoire, b un nombre ordinal de la personne. Les paires sont rangées selon a. Cette fonction fait la même chose que "zufall1":

```
zufall[{}]:={};
zufall[personen_List]:=
    Map[Last,
        Sort[Map[{Random[], #}&, personen],
             (#1[[1]] > #2[[1]])&
        ]];
zufall::usage="zufall[list] gibt eine zufällige \
           Permutation der Elemente der Liste \
           --- donne une permutation aléatoire \
           des éléments de la liste."
```

Einige Details zum Test:

• Quelques détails quant au test:


```
personen={a,b,c,d};
zufall[personen]

{Random[], #}&[personen]

Map[{Random[], #}&, personen]

Sort[Map[{Random[], #}&, personen],
           (#1[[1]] > #2[[1]])&]

Map[Last, Sort[Map[{Random[], #}&, personen],
           (#1[[1]] > #2[[1]])& ]]
```

Gib nun den Personen statt Namen Ordnungszahlen von 1 bis n. 1 bedeutet "am meisten bevorzugt" etc..

• Donne aux personnes au lieu de noms des nombres ordinaux de 1 jusqu'à n. 1 signifie "préféré le plus" etc..

```
n = 30;
personen = zufall[Range[n]]
```

Die Funktion "wahlVorzug" soll nun n Personen nehmen und diese in zufällige Reihenfolge bringen. Sie soll die ersten (n/e) (gerundet) Personen nehmen und den Wert der meist bevorzugten Person in dieser Menge in die Variable mmin speichern. Der Wert der ersten noch mehr bevorzugten Person, die dann folgt, soll in die Variable wahl gespeichert werden. Wenn wahl eine Zahl grösser gleich 1 enthält, soll wahlVorzug diese Zahl als Element einer Menge ausgeben. Andernfalls soll die leere Menge kommen.

• La fonction "wahlVorzug" prenne maintenant n personnes et les mette dans un ordre aléatoire. Qu'elle prenne les premières (n/e) (arrondi) personnes et qu'elle mémorise dans la variable mmin la valeur de la personne préférée de cet ensemble. La valeur de la personne préférée encore danvntage, qui suit, doit être mémorisée dans la variable wahl. Si wahl contient un nombre plus grand ou égal 1, wahl doit sortir ce nombre comme élément d'un ensemble. Autrement doit apparaître l'ensemble vide.

```
wahlVorzug[n_Integer]:=
        Module[{personen=zufall[Range[n]],
        m=Round[N[n/E]],
        mmin, wahl},
        mmin=Min[Take[personen,m]];
        wahl=Select[personen,(#<mmin)&];
        If[Length[wahl]>0,First[wahl],{}]];
anzahlPersonen=30;
anzahlLauf=50;
resultat=Table[wahlVorzug[anzahlPersonen],
        {anzahlLauf}]
```

Anzahl positive Simulationen mit Resultat 1, 2 oder 3:

• Nombre de limulations avec le résultat 1, 2 ou 3:

```
posSim1={Map[(N[Count[resultat,#]/anzahlLauf])&,
                                  {1,2,3}]}
```

Kürzer mit Vergleich mit (1/e):

• Plus brièvement avec la comparaison avec (1/e):

```
posSim1={(N[Count[resultat,#]/anzahlLauf
                        ])&/@{1,2,3},N[1/E]}
```

- **(b) Prozentualer Anteil der Läufe, die mit einem signifikanten Partner enden:**

€ Partie propositionelle des parcoours qui terminent par un partenaire significatif:

Studiert man den Output, so sieht man, dass meistens die Anzahl der Fälle, in denen die leere Menge herauskommt, sehr gering ist. Dann hat die Strategie keinen Erfolg. Falls die Strategie eine Zahl ergibt, so wollen wir von Erfolg insofern reden, dass dann eine kleine Nummer erwartet werden kann. Wir sagen dann, es hätte sich ein "signifikanter Partner" finden lassen. Wie oft kommt das prozentual vor?

• Si on étudie l'output, on voit que pour la plupart le nombre des cas où sort l'ensemble vide et très petit. Dans ces cas la stratégie n'a pas de succès. Si la stratégie donne un nombre, nous pouvons nous attendre à un nombre bas. Nous dirons qu'un "partenaire significatif" a pu être trouvé. Dans quel pourcentage cela arrive-t-il?

```
signifikant = 1- N[Count[resultat,{}]/50]
```

■ **c) Wie gut oder schlecht war die Auswahl nun?**
€ Combien le choix était donc bon ou mauvais?

Wir berechnen, in wievielen Läufen prozentual die Simulation mit einer Person endete, die unter "den favorisierten 10%" einzureihen ist.

• Nous calculons dans combien de parcours la simulation s'est terminée par une personne qu'on a pu placer parmi les "10 % facorisés".

```
signifikant10Prozent =
       N[Length[Select[resultat,
       (#<=Round[N[anzahlPersonen 0.1]])&]]/50]
```

■ **(d) Bei einer grösseren Laufzahl:**
€ Lors d'un nombre supérieur de parcours:

```
anzahlLauf=50;
resultat=Table[wahlVorzug[anzahlPersonen],
                  {anzahlLauf}];
posSim1={Map[(N[Count[resultat,#]/anzahlLauf])&,
                  {1,2,3}]};
signifikant = 1- N[Count[resultat,{}]/50];
signifikant10Prozent =
       N[Length[Select[resultat,
       (#<=Round[N[anzahlPersonen 0.1]])&]]/50];
Print["resultat = ", resultat];
Print["posSim1             = ", posSim1];
Print["signifikant         = ", signifikant];
Print["signifikant10Prozent = ",
                           signifikant10Prozent]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Kurs € Cours

13. Fallstricke und "Debugging" (Fehler eliminieren)
€ Embûches et "Debugging" (éliminer les erreurs)

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 13 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 13.
WIR94/99

■ 13.1. Fehlermeldungen
€ Annonces d'erreurs

Beispiel: • Exemples:

```
1 + 2 = 3
```

Hier kommt eine Fehlermeldung. Die Fehlermeldungen sind im Technical Report Number 9 ff von Wolfram Research beschrieben. Achtung! Orthographische, syntaktische Fehler (programmlogische Fehler, Durchbrechung der Programmierregeln) können entdeckt werden, jedoch nicht semantische Fehler (falsche Interpretation der eigenen Absicht in der Niederschrift).

• Voici une annonce d'erreurr.(Le annonces d'erreurs sont décrites dans le Technical Report Number 9 ff de Wolfram Research.) Attention! Erreurs d'orthographe ou de syntaxe (erreurs concernant la logique de programmation ou les règles de programmation) peuvent être découvertes, tandis que les erreurs sémantiques (fausse interprétation de la propre intention dans le manscrit) ne peuvent pas.

Beispiel: • Esemple:

```
2 - )5 + 7
```

Hinweis zum Notebook Front End: Mit (<Command>-B (B gross) --- NeXT) <Shift Control> B (B gross) ist eine Klammerüberprüfung möglich. Probiere das aus (setze den Cursor in die Lücke und verwende (<Command>-B (B gross) --- NeXT) <Shift Control> B (B gross)):

• Indication au sujet de Notebook Front End: Avec (<Command>-B (B grand) --- NeXT) <Shift Control> B (B grand) il est possible de faire un examin des parenthèses. Essaie (met le cursor dans l'espace vide et emploie (<Command>-B (B grand) --- NeXT) <Shift Control> B (B grand)):

```
((([    [cjklf]r]}}]
```

■ 13.2. Fehler durch vordefinierte Variablen
€ Erreur par des variables définies d'avance

Beispiel: • Exemple:

Wird einer Variablen ein Wert zugewiesen, so wird auf den Wert und nicht auf die Variable gegriffen:
• Quand une valeur est assignée à une variable, c'est la valeur qui est prise et non pas la variable:

```
t = 5;
Print["t = ", t];
Integrate[Sin[t], t]
```

Abhilfe: Z.B. mit lokalen Variablen arbeiten.
• Secours: P.ex. travailler avec des variables locales.

■ 13.3. Unvollständige Befehle
€ Ordres incomplets

■ 13.3.1.Fehlende oder nicht korrekte Namen
€ Noms qui manquent ou qui ne sont pas corrects.

Studiere: • Etudie:

```
D[x^3]

D[x^3, x]

D[x^3, y]

D[x^3, 1]

D[x^3 Sin[y], x, y]
```

D[x^3] ergibt offenbar die 0-te Ableitung von x^3.
• D[x^3] donne apparemment la 0-ième dérivation de x^3.

■ 13.3.2.Fehlende oder nicht korrekte Punktuation
€ Ponctuation qui manque ou qui n'est pas correcte

Studiere: • Etudie:

```
Clear[f];
f[x_] := If[x > 0,  x -x];
{f[5], f[-5]}
```

Mit Komma:
• Avec virgule:


```
Clear[f];
f[x_] := If[x > 0,  x, -x];
{f[5], f[-5]}
```

Studiere: • Etudie:

```
Clear[f];
f[x_, y_] := xy;
f[2, 3]

Clear[f];
f[x_, y_] := x y;
f[2, 3]

D[x^3 Sin[y], x, y]
```

■ 13.4. Falsche Klammerung
€ Erreurs de parenthèses

Beispiel: • Exemple:

```
Clear[g]

g[0]

g(0)

g(a)
```

■ 13.5. Verwechslung von Zeilen- und Spaltenvektor
€ Confusion entre vecteurs de ligne et vecteur de colonnes

■ Skalarprodukt
€ Produit scalaire

```
{a, b, c}.{e, f, g}

{a, b, c}.{e, f, g, h}
```

■ Matrixprodukt
€ Produit de matrice

Uebersicht: • Vue d'ensemble:

```
{{a},{b},{c}} // MatrixForm

Transpose[{{a},{b},{c}}] // MatrixForm
```

Diverse Produkte:
• Produits divers:

```
{{a},{b},{c}} . {{e},{f},{g}} // MatrixForm
```

```
{{a},{b},{c}} . Transpose[{{e},{f},{g}}] //
                        MatrixForm

Transpose[{{a},{b},{c}}] . {{e},{f},{g}} //
                        MatrixForm

Transpose[{{a},{b},{c}}] .
    Transpose[{{e},{f},{g}}] // MatrixForm
```

Generierung: • Génération:

```
Table[u[[i]] v[[j]], {i, Length[u]}, {j,
                        Length[v]}]
```

Was weiss man über u und v?
• Que sait-on de u et v?

■ 13.6. Weiter mit "Return" statt "Enter"
€ Continuer par "Return" au lieu de "Entrer"

▪ **Beispiele € Exemple**

```
a = 5
    + 7
    + 9

a

a = 5 +
    7 +
    9

a
```

Was ist hier los?
• Que se passe-t-il ici?

■ 13.7. Probleme mit Wurzeln (Mehrdeutigkeiten)
€ Problèmes de racines (différentes acceptions, équivoques)

▪ **Beispiele**
€ Exemples

```
(x^2)^(0.5)

Sqrt[x^2]

Integrate[Sqrt[1 - Cos[x]^2], {x, 0, Pi/2}]

Integrate[Cos[x], {x, Pi, 2Pi}]

Integrate[Sqrt[1 - Sin[x]^2], {x, Pi, 2Pi}]
```


Was ist hier los?
• Que se passe-t-il ici?

```
??Sqrt

Sqrt[(-2)^2]
```

■ 13.8. Abarbeitungsreihenfolge
€ Qrdre de travail

Probiere: • Essaier:

```
s = 2 x /(5x - 7)

Plot[s, {x, 0, 4}];

Plot[Numerator[s], {x, 0, 4}];
```

Wieso der Fehler? Probiere:
• Pourquoi l'erreur? Essaie:

```
Plot[Evaluate[Numerator[s]], {x, 0, 4}];
```

■ 13.9. Ordnung von Regeln
€ Ordre de règles

Probiere und erkläre die Prioritäten! Beispiel:
• Essaie et explique les priorités!

```
Clear[f];
f[x_] := x^2;
f[y_] := y^3;

{f[x],f[y]}

??f
```

Beispiel: • Exemple:

```
Clear[ung];
ung[x_?OddQ ] := (ung[x] = 1);
ung[x_?EvenQ] := (ung[x] = 0);


Table[ung[x], {x,10}]

??ung
```

■ 13.10. "Protect" und Funktionen
€ "Protect" et fonctions

```
??Protect

??Unprotect
```

Probiere: • Essaie:

```
Integrate[f[x_], x_] := F[x]

Remove[f, F];
f/: Integrate[f[x_], x_] := F[x];
Integrate[f[x], x]
```

Probiere: • Essaie:

```
Pi = 3.1416

{Pi, Pi // N}

Unprotect[Pi];
Pi = 3.1416;
Pi

Protect[Pi];
Pi // N

??Pi

Unprotect[Pi];
Remove[Pi];
Pi // N

??Pi
```


■ 13.11. Debugging
€ Debugging

■ 13.11.1. Allgemeine Regeln
€ Règles générales

Beachte beim Programmieren:

Dokumentiere die eigenen Funktionen.
Beachte Gross- und Kleinschreibung.
Beachte Klammerung.
Beachte die Initialisierung von Variablen.
Beachte die Funktionennamen.
Teste die Statements.
Beachte die Argumente von Funktionen.
Beachte die Punktuation.
Kontrolliere, ob Befehle oder Zeichen vergessen worden sind.
Mache Tests in trivialen Fällen.
Teste den Code mit Testdaten.
Beachte das Zuladen von Packages.
Beachte die "Contexts" (Erklärung in Kap. 15.)
Schreibe die Programme so, dass ein anderer nicht viel Zeit benötigt, um sie zu lesen. Wer die eigenen Kriterien den andern aufzwingen will, ist nicht sehr teamfähig. Denke auch an die andern, obwohl Du ja für Dich alleine sicher sehr gescheit bist!

•

Considère en programmant:

Documente les propres fonctions.
Tiens compte des majuscules et minuscules.
Tiens compte des initiales des variables.
Tiens compte des noms et fonctions.
Examine et teste les statements.
Tiens compte des arguments de fonction.
Tiens compte de la ponctuation.
Contrôle si des ordres ou des signes ont été oubliés.
Fais des tests dans des cas triviaux.
Teste le code par des données aptes à cela.
Tiens compte des Packages utilisés.
Tiens compte des "Contexts". (Explication au chapitre 15.)
Ecris les programmes de façon qu'un autre personne ne perde pas beaucoup de temps à les lire. Qui veut imposer ses propres critères aux autres, n'est pas capable de travailler en groupe. Pense aux autres, bien que tu n'aies pas le problèmes en travaillant seul.

Studiere: • Etudie:

```
?On

?Off
```

```
?Print

?Debug

?Trace

?Input

?Interrupt
```

■ 13.11.2. Traceing
€ Traceing

Probiere aus: • Essaie:

```
Remove[f, g];
f[x_]:= x^3 + g[x];
g[x_]:= x - 14

On[f, g]

f[4]

Off[f, g]

f[4]
```

■ 13.11.3. Print
€ Print

Von 10.3.7:
• De 10.3.7:

```
median[liste_List]:=
    Block[{
            sl,
            len
          },
          len = Length[liste];
          Print["Länge der Liste: ", len];
          sl = Sort[liste];
          Print["Sortierte Liste: ", sl];
          Print["Median: "];
          If[
             OddQ[len],
             sl[[(len+1)/2]],
             (sl[[len/2]]+sl[[len/2+1]])/2
          ]
    ];
    median[{24, 32, 36, 47, 49, 52}]
```


■ 13.11.4. Debug (für Version 1.2)
€ Debug (pour version 1.2)

Beispiel: • Exemple:

```
?$Version

?$VersionNumber

$VersionNumber < 2

If[$VersionNumber < 2,
   Debug[For[n = 1, n < 4, ++n, Print[n^3]]],
   Print["Achtung: Bei Debug ab Version 2:
            Systemabsturz! \
   Attention: Debug dès version 2:
            système interrompu!"]]
```

■ 13.11.5. Trace
€ Trace

Studiere: • Etudie:

```
2 5 + 8 9

Trace[2 5 + 8 9]

Trace[2 5 + 8 9, TraceDepth -> 1]

Trace[2 5 + 8 9, Plus]

Trace[2 5 + 8 9, Times]

Trace[2 5 + 8 9, x_ y_]

Trace[2 5 + 8 9, TraceDepth -> 1]
```

■ 13.11.6. Input
€ Input

Studiere (obiges Beispiel abgeändert):
Falls während der Rechnung plötzlich ein Fenster geöffnet wird so kannst Du eine Variable abfragen. Schreibe z.B. "?len" ins Fenster und beobachte, was passiert!
•
Etudie (Exemple ci-dessus changé):
Si pendant le calcul il s'ouvre tout à coup un fenêtre, tu peux demander le centenu d'une variable. Ecris p.ex. "?len" dans la fenêtre et observe ce qui se passe!

```
median[liste_List]:=
    Block[{
            sl,
            len
          },
          len = Length[liste];
          sl = Sort[liste];
          Print[Input[]];
          If[
             OddQ[len],
             sl[[(len+1)/2]],
             (sl[[len/2]]+sl[[len/2+1]])/2
          ]
    ];
    median[{24, 32, 36, 47, 49, 52}]
```

■ 13.11.7. Interrupt
€ Inerrupt

Beispiel: n! soll berechnet werden. Die Rechnung soll wegen der Rechnungszeit aber nur durchgeführt werden für n <= 1000.

• Exemple: n! doit être calculé, mais à cause du temps de calcul on ne le calculera que pour des n >= 1000.

```
Remove[f];
f::gross = "Eingegebene Zahl grösser tausend. \
       Rechenzeit möglicherweise sehr lang.";
f[n_] := If[n <= 500, Print[n,"! = ", n!],
       f::gross]

f[10]

f[100]

f[500]

f[501]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Uebungen € Exercices

13. Fallstricke und "Debugging" (Fehler eliminieren)
€ Embûches et "Debugging" (éliminer les erreurs)

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 13 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 13.
WIR94/99

■ "Putzmaschine" einsetzen (hier sehr wichtig)
€ Employer la "machine de nettoyage" (tres important dans ce cas)

```
Remove["Global`@*"]
```

■ Versuche, in den folgenden Beispielen die Fehler zu finden!
€ Essaie de trouver les erreurs dans les exemples suivants!

■ Aufgabe 1 € Problème 1

```
3 Plus 4
```

(Ueberlege, wie "Plus" wirkt.)

• (Réfléchis aux effets de "Plus")

```
Plus[3, 4]
```

■ Aufgabe 2 € Problème 2

```
mean[x_] := Add[x]/Length[x];
mean[{1,2,3}]
```

Schreibe den Code richtig!

• Ecris le Code correctement!

```
?Plus

Remove[mean];
mean[{x__}] := Plus[x]/Length[{x}];
mean[{1,2,3}]
```

■ Aufgabe 3 € Problème 3

Wieso kommen keine Primzahlen?

- Pourquoi il n'apparaît pas de nombre premier?

```
Select[{2, 3, 7, 9, 11, 12}, PrimeQ[x_]]
```

Schreibe den Code richtig!

- Ecris le code correctement!

```
?Select

Trace[Select[{2, 3, 7, 9, 11, 12}, PrimeQ[x_]]]

Select[{2, 3, 7, 9, 11, 12}, PrimeQ]
```

■ Aufgabe 4 € Problème 4

Berechne das Integral!

- Calcule l'intégral!

```
Integral[xCos(x), (x, 0, pi)]
```

Schreibe den Code richtig!

- Ecris le code correctement!

```
Integrate[x Cos[x], {x, 0, Pi}]
```

■ Aufgabe 5 € Problème 5

Mit der folgenden Funktion sollen 6 Lottozahlen ausgewählt werden. Wieso kommt nur eine?

- Avec la fonction suivante on choisit 6 nombres de loto. Pourquoi il n'en vient qu'un?

```
machLotto:=
    Block[{
        n = 1
          },
          While[n <= 6;
            Print[
                Random[Integer, {0, 49}]
            ];
            n++
          ]
    ];
machLotto
```


Schreibe den Code richtig!

- Ecris le code correctement!

```
machLotto:=
    Block[{
        n = 1
          },
          While[n <= 6,
            Print[
                Random[Integer, {0, 49}]
            ];
            n++
          ]
    ];
machLotto
```

■ Aufgabe 6 € Problème 6

Studiere die folgenden Funktionen:

- Etudie les fonctions suivantes:

```
Remove[aoi, eip];
aoi[f_[x_], min_, max_] := NIntegrate[f[x],
                                {x, min, max}];
eip[x_               ] := x^2 + 5 Exp[x];
aoi[eip[t], 5, 10]
```

Wieso kommt nicht ein numerischer Wert Hier einige Studien und Lösungsmöglichkeiten:

- Pourquoi il n'apparaît pas une valeur numérique? Voici quelques études et quelques possibilités de solution:

```
?aoi

On[aoi, eip]

aoi[eip[t], 5, 10]

Remove[aoi, eip];
On[aoi, eip];
aoi[f_[x_], y_, min_, max_] := NIntegrate[f[x],
                                   {y, min, max}];
eip[z_               ] := z^2 + 5 Exp[z];
aoi[eip[t], 5, 10]

Remove[aoi, eip];
On[aoi, eip];
aoi[f_, y_, min_, max_] := NIntegrate[f,
                                {y, min, max}];
eip := x^2 + 5 Exp[x];
aoi[eip, t, 5, 10]

aoi[eip, x, 5, 10]

Remove[aoi, eip];
On[aoi, eip];
aoi[f_, x_, min_, max_] := NIntegrate[f,
                                {x, min, max}];
eip[y_               ] := y^2 + 5 Exp[y];
aoi[eip[t], t, 5, 10]
```

■ Aufgabe 7 € Problème 7

Schreibe die folgende Funktion kürzer und besser:

• Ecris la fonction suivante d'une daçon plus courte et meilleure:

```
Remove[f];
f[g_, x_List]:=
    Block[{
        l = {},
        n = 1
    },
    Do[
        l = Join[l, {g[x[[i]]]}],
        {i, Length[x]}
    ];
    l
];
f[g, {a, b, c}]
```

Einfacher: • Plus simplement:

```
Clear[f];
f[g_, x_List] := Map[g, x];
f[g, {a, b, c}]
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Kurs € Cours

14. Input und Output
€ Input et output

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 14 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 14.
WIR94/99

Problem: Datenimport und Datenexport in und von *Mathematica*.

• Problème: Importation et exportation de données dans et de *Mathematica*.

■ 14.1. Input
€ Input

■ 14.1.1. Dateneingabe via Tastatur während des Programmlauf
€ Entrée des données par les touches pendant que le programme marche

Studiere: • Etudie:

```
?Input
```

```
?InputString
```

```
?BaseForm
```

```
BaseForm[15,2]+BaseForm[17,2]
```

```
benutzerInput = Input[
"Schreibe eine Zahl zwischen 1 und 10: \
Ecris un nombre entre 1 et 10:"]
```

Achtung: Das nachfolgende Programm bricht erst ab, wenn keine Integer-Zahl eingegeben wird! Probiere:

• Attention: Le programme suimant s'interrompt seulement si on n'entre pas le nombre Integer! Essaie:

```
dualUmwandlung:=
    Module[{
            benutzerInput},
            While[
                IntegerQ[
                    userInput = Input[
    "Schreibe eine Zahl zwischen 1 und 10: \
    Ecris un nombre entre 1 et 10:"]
                        ],
                Print[userInput, " = ",
                        BaseForm[userInput, 2]]
            ]
    ];

dualUmwandlung
```

■ 14.1.2. Datenimport von einem File
€ Importation de données d'un file

Achtung: Achte auf das zu deinem Betriebssystem lauffähige Beispiel!
Der Umgang mit externen Dateien von *Mathematica* aus hat ab Version 3.0 eine grosse Veränderung erfaren!
€ Attention: Prends seulement les exemples qui tournent sur ton propre système!
Le traitement des fichiers externs depuis *Mathematica* a beaucoup changé depuis la version 3.0!

Beispiel:
(Das File AAAdaten) befindet sich wahrscheinlich in Deinem Verzeichnis
C:\Programme\WolframResearch*Mathematica*\3.0 (win98). - Sonst musst Du Deinen mir nicht bekannten Pfad suchen!)
• Exemple:
(Le file AAAdanten.nb (aadaten.ma) se trouve probablementdans le directoire
C:\Programme\WolframResearch*Mathematica*\3.0 (win98). - Si non tu dois chercher ton "path" que je ne connais pas!)

Mathematica 3.02, Windows98
Beispiel: neues Verzeichnis erstellen:
• Faire un nouveau directoire:

```
CreateDirectory["C:\work"]
```

```
C:\workNew
```

```
x = Sin[3] // N >> AAADaten
```

```
<< AAADaten
```

```
0.1411200080598672
```

Beispiel: Erstelle ein Verzeichnis C:\work (falls nicht vorhanden).
(Das File AABDaten befindet sich in Deinem Verzeichnis "work" (win98), (aadaten.ma: "Mate" (win3.1, 95....), "WirzMathematicaFiles/Daten" (NeXT) usw.) . - Sonst musst Du Deinen mir nicht bekannten Pfad angeben!)

• Exemple: Ouvre un directoire C:\work (si cela n'existe pas encore).
(Le file AABDaten se trouve dans le directoire "work" (win98), (aadaten.ma: "Mate" (win3.1, 95....), "WirzMathematicaFiles/Daten" (NeXT) etc.). - Si non tu dois m'indiquer ton "path" que je ne connais pas!)

Schreibe Daten ins File C:\work\AABDaten:
• Ecrire des données dans le fichier C:\work\AABDaten:

```
"3 4 5 1 2 3 6 6.3 7.8 4 r s t hallo
 7777 6666 4.321 1.11111, {1,2,3,4}" >> c:\work\AABDaten;
```

```
{"Cos[3]//N", Cos[3] // N} >> c:\work\AACDaten;
```

```
{3 4 5 1 2 3 6 6, "3 4 5 1 2 3 6 6",
 {3 4 5 1 2 3 6 6}} >> c:\work\AADDaten;
```

Lese das File C:\work\AABDaten:
• Lire le fichier C:\work\AABDaten:

```
<< c:\work\AABDaten
```

```
!! c:\work\AABDaten
```

win98:
Erstelle nötigenfalles eine Textdatei "c:\work\AAEDaten.txt" mit Inhalt:
• Si nécessaire composer un fichier "c:\work\AAEDaten.txt" avec le contenu:
1 2 3 4 5 6 7
1.1 1.2 1.3 1.4
11 12 13 14 115
a b c d e f g
Sin[5]\\N hallo
"hallo"

Erstelle ebenso eine Textdatei "c:\work\AAFDaten.txt" mit Inhalt:
• Composer aussi un fichier "c:\work\AAEDaten.txt" avec le contenu:
1 2 3 4 5 6 7
1.1 1.2 1.3 1.4
11 12 13 14 115

Inhalt anschauen:
• Regarder le contenu:

```
!! c:\work\AAEDaten.txt
```

```
!!f:\Mathe/Daten/aadaten0 (* win3.1 etc. *)
```

File einlesen und ausführen:

```
<< c:\work\AAEDaten.txt
```

```
<<f:\Mathe/Daten/aadaten1 (* win3.1 etc. *)
```

Was ist los gewesen? Schauen wir weiter:
• Que s'est-il passé? Continuons d'observer:

```
?<<

?Get

!!c:\work\AAEDaten.txt

!!f:\Mathe/Daten/aadaten1 (* win3.1 etc. *)

<<c:\work\AAEDaten.txt

<<f:\Mathe/Daten/aadaten1 (* win3.1 etc. *)

Get["c:\work\AAEDaten.txt"]

Get["f:\Mathe/Daten/aadaten1"]
                          (* win3.1 etc. *)

1 7 3 4 2 9 (*Produkt! *)
```

Spezifisch einlesen:

- Lire spécifiquement:

```
?ReadList

ReadList["c:\work\AAFDaten.txt", Real]

ReadList["f:\Mathe/Daten/aadaten0", Real]
                          (* win3.1 etc. *)

?Real

ReadList["c:\work\AAFDaten.txt", Number]

ReadList["f:\Mathe/Daten/aadaten0", Number]
                          (* win3.1 etc. *)

?Number

ReadList["c:\work\AAFDaten.txt", Byte]

ReadList["f:\Mathe/Daten/aadaten0", Byte]
                          (* win3.1 etc. *)

?Byte

ReadList["c:\work\AAFDaten.txt",
                       Character]

ReadList["f:\Mathe/Daten/aadaten0",
              Character]  (* win3.1 etc. *)

?Character

ReadList["c:\work\AAEDaten.txt", String]

ReadList["f:\Mathe/Daten/aadaten0", String]
                          (* win3.1 etc. *)

?String

ReadList["c:\work\AAEDaten.txt", Record]
```



```
ReadList["f:\Mathe/Daten/aadaten0", Record]
                          (* win3.1 etc. *)

?Record

ReadList["c:\work\AAEDaten.txt", Word]

ReadList["f:\Mathe/Daten/aadaten0", Word]
                          (* win3.1 etc. *)

?Word

ReadList["f:\Mathe/Daten/aadaten1", Real]
                          (* win3.1 etc. *)

ReadList["f:\Mathe/Daten/aadaten1", Number]
                          (* win3.1 etc. *)

ReadList["f:\Mathe/Daten/aadaten2", Number]
                          (* win3.1 etc. *)

!!f:\Mathe/Daten/aadaten2   (* win3.1 etc. *)

ReadList["c:\work\AAFDaten.txt",
              {Number, Number}]

ReadList["f:\Mathe/Daten/aadaten1",
       {Number, Number}]   (* win3.1 etc. *)

ReadList["c:\work\AAFDaten.txt",
       {Number, Number, Number, Number}]

ReadList["f:\Mathe/Daten/aadaten1",
       {Number, Number, Number, Number}]
                          (* win3.1 etc. *)

ReadList["c:\work\AAFDaten.txt",
              Cos[Number]]

ReadList["f:\Mathe/Daten/aadaten1",
       Cos[Number]]       (* win3.1 etc. *)

ReadList["c:\work\AAFDaten.txt",
              Cos[Number]] // N

ReadList["f:\Mathe/Daten/aadaten1",
       Cos[Number]] // N   (* win3.1 etc. *)

!!f:\Mathe/Daten/aadaten3   (* win3.1 etc. *)

ReadList["c:\work\AAFDaten.txt",
{String, Number, Number}]

ReadList["f:\Mathe/Daten/aadaten3",
{String, Number, Number}]   (* win3.1 etc. *)

!!f:\Mathe/Daten/aadaten4   (* win3.1 etc. *)

ReadList["c:\work\AAFDaten.txt",
{Number, Number, String}]
```

```
ReadList["f:\Mathe/Daten/aadaten4",
{Number, Number, String}]   (* win3.1 etc. *)

!!f:\Mathe/Daten/aadaten5   (* win3.1 etc. *)

ReadList["f:\Mathe/Daten/aadaten5",
{Number, Number, String}]   (* win3.1 etc. *)
```

Probiere eigene files!

- Essaie tes propres fichiers!

■ 14.1.3. Einlesen von Records und Words
€ Lire Records et Words

Studiere die "Befehle":

- Etudie les "ordres":

```
?*Record*

?RecordLists

?RecordSeparators

?WordSeparators

?NullRecords

?NullWords

?TokenWords
```

Studiere die Anwendungen:

- Etudie les applications:

```
!!c:\work\AAGDaten.txt

!!f:\Mathe/Daten/spsheet0 (* win3.1 etc. *)

ReadList["c:\work\AAGDaten.txt",
 {Number, Number, Number}]

ReadList["f:\Mathe/Daten/spsheet0",
 {Number, Number, Number}](* win3.1 etc. *)

ReadList["c:\work\AAGDaten.txt",
 Word, RecordLists -> True]

ReadList["f:\Mathe/Daten/spsheet0",
 Word, RecordLists ->True] (* win3.1 etc. *)

ReadList["c:\work\AAGDaten.txt",  Word]

ReadList["f:\Mathe/Daten/spsheet0",
 Word]                     (* win3.1 etc. *)

ReadList["c:\work\AAGDaten.txt",
          RecordLists -> True]
```



```
ReadList["f:\Mathe/Daten/spsheet0",
       RecordLists ->True] (* win3.1 etc. *)

Clear[x];
x = ReadList["f:\Mathe/Daten/spsheet0",
        Word,
        WordSeparators -> {"\t"},
        RecordLists ->True] (* win3.1 etc. *)

Clear[x];
x = ReadList["c:\work\AAGDaten.txt",
  Word,
  WordSeparators -> {"\t"},
  RecordLists -> True]

ToExpression[x]

?ToExpression

FullForm[x]

ReadList["f:\Mathe/Daten/spsheet1",
        {Word, Number, Number, Number},
        WordSeparators -> {" "},
        RecordLists ->True] (* win3.1 etc. *)

ReadList["c:\work\AAGDaten.txt",
  {Word, Number, Number, Number},
  WordSeparators -> {" "},
  RecordLists -> True]

FullForm[%]
```

■ 14.1.3. Ein längeres Beispiel
€ Un exemple plus long

Zuerst definieren wir Funktionen. Damit kŏnnen wir obigen Befehl "nachbilden":

- D'abord nous définissons des fonctions. Avec cela nous pouvons "imiter" l'ordre ci-dessus.

```
Clear[tabSeparate, tabSep];
tabSeparate[str_String]:= Apply[tabSep,
        Characters[str]];
tabSep[x__String,"\t",y___String]:=
        Flatten[{tabSep[x],tabSep[y]},1];
tabSep[x___String]:=
        ToExpression[StringJoin[x]]/;
        FreeQ[{x},"\t"];
ReadList["c:\work\AAGDaten.txt",
        tabSeparate[String]]
```

```
Clear[tabSeparate, tabSep];
tabSeparate[str_String]:=Apply[tabSep,
       Characters[str]];
tabSep[x__String,"\t",y___String]:=
       Flatten[{tabSep[x],tabSep[y]},1];
tabSep[x___String]:=
       ToExpression[StringJoin[x]]/;
       FreeQ[{x},"\t"];
ReadList["f:\Mathe/Daten/spsheet1",
       tabSeparate[String]](* win3.1 etc. *)

?Characters

?FreeQ
```

Achtung! *Mathematica* ist nicht für Datamanagement und Datamanipulation gebaut. Man kann Daten in speziellen Datamanagement-Sprachen oder Programmen (low-level- Programmierung) viel schneller parsen (einlesen und interpretieren) sowie manipulieren. Dies ist bei grösseren Datenmengen wesentlich. Man denke z. B. an (1000 x 1000)-Matrizen in der Statistik, z.B. bei 1000 Beobachtungen von 1000 Variablen.

• Attention: *Mathematica* n'est pas conçu pour le "management" et pour la manipulation des données. On peut lire et intérpréter ("parser") de même que manipuler beaucoup plus rapidement les données à l'aide de langages "datamanagement" ou programmes (programmation low-level). Cela est essentiel pour des quantités plus importantes de données. On pense p.ex. aux matrices (1000 x 1000) dans la statistique, p.ex. 1000 observations da 1000 variables.

■ 14.1.4. Externe Kommandos bis exklusive Version 3.0 (für WIN98 nicht zu empfehlen)
€ Commandements extérieurs jusqu'à version 3.0, exclusive (pour WIN98 pas recommandé)

Mit "!Kommando" kann ein externes Kommando ausgeführt werden. Leider haben wir auf NeXT keinen ModulaII-Compiler oder Java-Compiler, so dass wir selbst solche Kommandos bauen könnten. Eine andere Programmiersprache, die den Zweck erfüllen würde, wird an der Schule momentan nicht unterrichtet. Hier ist für NeXT das Programm "aa_ascii", das Umlaute in TeX-Format übersetzt. Studiere das Beispiel:

• Par "commandement !" on peut exécuter un commandement extérieur. Malheureusement nous n'avons pas de compilateur pour ModulaII ou Jave sur NeXT pour pouvoir construire de tels commandements. Une autre langue de programmation qui serve pour le même but n'est actuellement pas enseigné à cette école. Ici on utilise pour NeXT le programme "aa_ascii" qui transforme les "Umlaute" (métaphonie, voyelle infléchie) de l'allemande dans le format TeX. Etudie l'exemple:

```
!!f:\Mathe/Daten/spsheet2

!f:\Mathe/Daten/aa_ascii
       f:\Mathe/Daten/spsheet2
       f:\Mathe/Daten/tex -tex;
ReadList["f:\Mathe/Daten/tex",
       {Word, Number, Number, Number},
        WordSeparators -> {" "},
        RecordLists ->True]
```

Das "!Kommando" kann auch in ReadList stehen, falls es sich auf eine Operation auf einem File bezieht (und nicht zwei Files aufruft)! Folgendes geht nicht:

• Le "commandement !" peut se trouver aussi dans ReadList, si il se rapporte à une opération sur un fichier (et n'appelle pas deux fichiers)! La chose suivante ne va pas:


```
ReadList["!f:\Mathe/Daten/aa_ascii
       f:\Mathe/Daten/spsheet2
       f:\Mathe/Daten/tex -tex ",
       {Word, Number, Number, Number},
        WordSeparators -> {" "},
        RecordLists ->True]
```

■ Externe Kommandos ab Version 3.0
€ Commandements extérieurs depuis version 3.0

Verwendung externer Programme für Datenmanipulationen von *Mathematica* aus mit Hilfe der Befehle "Install", "Links", "MLPut...",
"MLGet...", u.s.w. ..... .
Momentan sind keine Programme vorhanden, die problemlos auf irgendwelchen hier unbekannten Systemen laufen.
Daher wird aif ein konkretes Beispiel verzichtet. Probiere daher mit eigenen Programmen.

• Utiliser des Programmes externes pour manipuler des données depuis *Mathematica* à l'aide des ordres "Install", "Links", "MLPut...",
"MLGet...", etc. .... .
Dans ce moment moment nous n'avons pas de programmes qui tournent sans problèmes sur n'importe quel système qu'on ne connait pas ici dans ce moment. C'est pourquoi nous renonceons à présenter un exemple concrèt. Essaie avec tes propres programmes.

■ 14.2. Datenexport
€ Exportation de données

■ 14.2.1. Zeichenweise in ein File schreiben
€ Ecrire les signes l'un après l'autre dans un fichier

■ Beispiel
€ Exemple

Zuerst Daten erzeugen, die exportiert werden sollen:

• D'abord générer les données qui doivent être exportées:

```
Clear[daten];
daten = Table[Random[Integer,{1,9}],{2},{6}]
```

Studiere die Befehle:

• Etudie les ordres:

```
?>>

?Put

??WriteString

?OpenWrite

?Close
```

```
?Scan

?Rest
```

Zum Exportieren wollen wir eine Funktion definieren, die die Generierte in der Art Matrix "Zeilenvektor auf Zeile ... (Record) im File" schreibt:
• Nous allons définir une fonction pour exporter, qui écrit la générée dans la matrice dans le sens "vecteur de ligne sur ligne .... (Record) dans le fichier":

```
Remove[schreibMatrix];
Off[neufile];
schreibMatrix[fileName_String, data_List] :=
    Block[{ neufile = OpenWrite[fileName]},
        Scan[(WriteString[neufile,First[#]];
                  Scan[WriteString[neufile,
                  "\t",#]&,Rest[#]];
             WriteString[neufile,"\n"])&,
             data];
        Close[neufile]];

schreibMatrix["C:\work\AAHDaten", daten]

schreibMatrix["f:\Mathe/Daten/neuDaten, daten] (* win3.1 etc. *)

!!C:\work\AAHDaten

!!f:\Mathe/Daten/neuDaten (* win3.1 etc. *)

?schreibMatrix
```

■ 14.2.2. Abspeichern von Definitionen
€ Mémoriser des définitions

■ Beispiel
€ Exemple

Studiere: • Etudie:

```
?Save

f[x_] := Sin[x^2] / (E^(-x) Cos[x]);
Save["C:\work\AA_f", f]

f[x_] := Sin[x^2]/(E^(-x) Cos[x]);
Save["f:\Mathe/Daten/AAA_f", f]
                        (* win3.1 etc. *)

!!C:\work\AA_f

!!f:\Mathe/Daten/AAA_f  (* win3.1 etc. *)
```


■ 14.2.3.Abspeichern von Output
€ Mémoriser des définitions

Studiere: • Etudie:

```
?>

?>>

?>>>

?Put
```

Probiere aus: • Essaie:

```
Table[f[n], {n, 20}] >>
 C:\work\AAneuDaten

Table[f[n], {n, 20}] >>
    f:\Mathe/Daten/neuDaten  (* win3.1 etc. *)

!!C:\work\AAneuDaten

!!f:\Mathe/Daten/neuDaten    (* win3.1 etc. *)

Put[Table[f[n], {n, 20}], Sin[x], "Hallo",
 "C:\work\AAneuDaten"]

Put[Table[f[n], {n, 20}], Sin[x], "Hallo",
 "f:\Mathe/Daten/neuDaten"]  (* win3.1 etc. *)

!!C:\work\AAneuDaten

!!f:\Mathe/Daten/neuDaten    (* win3.1 etc. *)
```

Was ist mit den Daten von 14.2.3 passiert?
Aue s'est-il passé avec les données de 14.2.3?

```
2.^0.5

2.^0.5 >> C:\work\AAneuDaten

2.^0.5 >> f:\Mathe/Daten/neuDaten
                          (* win3.1 etc. *)

!!C:\work\AAneuDaten

!!f:\Mathe/Daten/neuDaten   (* win3.1 etc. *)
```

Wieso ist das abgespeicherte Resultat genauer als das auf dem Schirm ausgegebene Resultat?
• Pourquoi le résultat mémorisé est-il plus exact que le résultat sorti sur l'écran?

■ 14.3. Manipulation von Strings
€ Manipulation de Strings

Studiere die folgenden Befehle:
• Etudie les ordres suivants:

```
?Read

?ReadList

?OpenAppend

?OpenRead

?StringDrop

?StringInsert

?StringJoin

?StringReplace

?StringReverse

?StringTake

?StringLength

?StringPosition

?StringMatchQ

?StringQ

?StringToStream

?WriteString

?ToString
```

Probiere aus: • Essaie:

```
Table[StringJoin["daten.", ToString[i]],{i,6}]
```

■ 14.4. Eingabe- und Ausgabeform
€ Formes d'entrée et de sortie

Studiere die folgenden Befehle:
• Etudie les ordres suivants:

```
?InputForm

?OutputForm
```

Beispiele: • Exemples:


```
Clear[a];
a = x^5/x^2 y +x^y^5

InputForm[a]

OutputForm[a]

a >> C:\work\AAneuDaten

a >> f:\Mathe/Daten/neuDaten  (* win3.1 etc. *)

!!C:\work\AAneuDaten

!!f:\Mathe/Daten/neuDaten     (* win3.1 etc. *)

Read["C:\work\AAneuDaten"]

Read["f:\Mathe/Daten/neuDaten"](* win3.1 etc. *)

b = Read["C:\work\AAneuDaten"]

b = Read["f:\Mathe/Daten/neuDaten"]
                              (* win3.1 etc. *)

b

!!C:\work\AAneuDaten

!!f:\Mathe/Daten/neuDaten     (* win3.1 etc. *)

Close["C:\work\AAneuDaten"]

Close["f:\Mathe/Daten/neuDaten"]
                              (* win3.1 etc. *)

b = Read["C:\work\AAneuDaten"]

b = Read["f:\Mathe/Daten/neuDaten"]
                              (* win3.1 etc. *)

Close["C:\work\AAneuDaten"]

Close["f:\Mathe/Daten/neuDaten"]
                              (* win3.1 etc. *)
```

■ 14.5. Uebersetzen von Ausdrücken in andere Programmiersprachen und Manipulation von Sourcecode
€ Traduire des expressions en d'autres languages de programmation et manipulation de sourcecode

■ Uebersetzungen
€ Tradcuction

Studiere die folgenden Befehle:
• Etudie les ordres suivants:

```
?CForm

?FortranForm

?TeXForm

?Splice
```

Beispiele: • Exemple:

```
Clear[a];
a = x^5/z^2 y +x^y^5 +
                ( x^2 Sin[x^5/z^2 y +x^y^5]);

CForm[a]

FortranForm[a]

TeXForm[a]
```

- **Manipulation von Sourcecode**
  € Manipulation de sourcecode

Im folgenden Beispiel wird gezeigt, wie in den Source-Code eines externen Programms (z.B. in C) direkt Mathematica-Ausdröcke eingesetzt und dann von Mathematica übersetzt werden kőnnen. Der externe Programmteil befindet sich im File AAA.mc. Das übersetzte Programm wird in AAA.c gespeichert!!!!!!!
• Dans l'exemple suivant on montre aomme on peut mettre des expressions de *Mathematica* directement dans le Source-Code d'un autre programme extérieur (p.ex. dand C), ensuite ses expressions peuvent être traduites par *Mathematica.* La partie extérieur du programme se trouve dans le fichier AAA.mx resp. AAA.mc. Le programme traduit est mémorisé dans AAA.x resp. AAA.c !!!!!!!

```
Sin[3] + Cos[3] >> C:\work\AAA.mx

!!C:\work\AAA.mx

!!f:\Mathe/Daten/AAA.mc     (* win3.1 etc. *)

Splice["C:\work\AAA.mx"]

Splice["f:\Mathe/Daten/AAA.mc"]
                            (* win3.1 etc. *)

!!C:\work\AAA.x

!!f:\Mathe/Daten/AAA.c      (* win3.1 etc. *)
```


■ 14.6. Formate
€ Formats

- **Allgemeines**
  € Généralités

Studiere die folgenden Befehle:
• Etudie l'ordre suivant:

```
?Subscripted

?Superscript

?SequenceForm
```

Probiere aus: • Etudie:

```
Subscripted[abc[145]]

SequenceForm[uvw, Superscript[251]]
```

- **Ein Anwendungsbeispiel**
  € Un exemple d'application

```
Remove[a, n, m];
Format[a[n_, m_]] := Subscripted[a[n, m]];
Array[a, {2, 2}] // MatrixForm
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Uebungen € Exercices

14. Input und Output
€ Input et output

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 14 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 14.
WIR94/99

■ Aufgabe 1 € Problème 1

Studiere die nachfolgenden Beispiele und den Befehl Input:
• Etudie les exemples suivants et l'ordre Input

```
Remove[MeinReadList];
MeinReadList[fileName_String,
        thing_:Expression]:=
    Module[{file, expr, list = {}},
        file = OpenRead[fileName];
        If[ file === $Failed, Return[list] ];
        While[ True,
            expr = Read[file, thing];
            If[ expr === EndOfFile, Break[] ];
            AppendTo[list, expr]
        ];
        Close[file];
        list
    ];
```

```
MeinReadList["C:\work\AAFdaten.txt", Number]

MeinReadList["f:\Mathe/Daten/AAAdaten", Number]              (* win3.1 etc. *)
```



```
Remove[MeinReadList];
MeinReadList[thing_:Expression] :=
    Module[{file, fileName, expr, list = {}},
        fileName = Input["Bitte einzulesende
        Datei in Anführungszeichen angeben.
        (Das File C:\work\AAFdaten.txt ...
        sollte vorhanden sein.) -- S.v.p
        enterer le fichier à lire entre
        parenthèses.(Le fichier
        C:\work\AAFdaten.txt doit exister.)"];
        file = OpenRead[fileName];
        If[ file === $Failed, Return[list] ];
        While[ True,
            expr = Read[file, thing];
            If[ expr === EndOfFile, Break[] ];
            AppendTo[list, expr]
        ];
        Close[file];
        list
    ];
    MeinReadList[Number]
```

■ Aufgabe 2 € Problème 2

Schreibe eine Funktion, mit der sich das Multiplizieren der Zahlen von 1 bis 12 im Kopf trainieren und korrigieren lässt!
• Ecris une fonction par laquelle on peut exercer et corriger le calcul mental de la multiplication des nombres de 1 à 12 !

- **Beispiel einer Lösung:**
  € Exemple d'une solution:

```
Remove[mult];
mult:=
    Module[{faktor1, faktor2,
            korrektur, resultat},

            Clear[resultat];
            resultat = ".";

      While[ resultat != "y" || resultat !
            = "Y" || resultat != "<y>" ||
            resultat != "<Y>", faktor1 =
            Random[Integer,{1,12}]; faktor2
            = Random[Integer,{1,12}];
            korrektur = faktor1 faktor2;
            Print["Was gibt ",faktor1,
            " mal ",faktor2,"?"];
            resultat = Input[
    "Bitte Resultat eingeben, oder beenden
    mit dem Buchstaben y. -- S.V.P entrer le
    résultat ou terminer à l'aide de y."];
         If[ resultat == korrektur,
               Print[resultat,
               " ist richtig, bravo! ---
               est juste, bravissimo!"],
               Print[resultat,
               "? - Richtig ist ---
               résultat juste:",korrektur]
            ]
          ];
          Print["Ende"]
    ];
mult

mult
```

■ Aufgabe 3 € Problème 3

Uebersetze die folgenden Ausdrücke mit Hilfe von *Mathematica* in C und in Fortran:
• Traduis les expressions suivantes à l'aide de *Mathematica* dans C et dans Fortran:
a) D[z Exp[z]
b) a^x
c) Sin[Pi/5]

Lösung: • Solution:

```
Map[{CForm[#], FortranForm[#]}&,
{D[z Exp[z], z], a^x, Sin[Pi/5]}]
```


■ Aufgabe 4 € Problème 4

Schreibe ein C-Programm, ein Fortran-Programm oder ein TeX-Programm, in dem Splice zur Anwendung kommt.
• Ecris un programme C, un programme Fortran ou un Programme TeX, dans lequel on applique Splice

- **Ein Beispiel: Manipulation von Sourcecode bei TeX**
  Un exemple: Manipulation de Sourcecode dans TeX

Im folgenden Beispiel wird gezeigt, wie in den Source-Code eines externen Programms (hier in TeX) direkt *Mathematica*-Ausdrücke eingesetzt und dann von *Mathematica* übersetzt werden kőnnen. Der externe Programmteil befindet sich im File C:\work\AAATeX.txt. Das übersetzte Programm wird in C:\work\AAA.tex gespeichert !!!!!!!
• Dans l'exemple suivant on montre comme on peut mettre directement des expressions de *Mathematica* dans la sorce-code d'un programme dxtérieur (ici dans TeX), ensuite elles peuvent être traduites par *Mathematica*. La partie extérieure du programme se trouve dans le fichier C:\work\AAATeX.txt. Le programme traduit dans le fichier C:\work\AAA.tex !!!!!!!!

```
TeXForm[Integrate[Sin[2 x + 3], x]; Sin[x] == 4 / Tan[y]; Print["Hallo!"]] >>
 C:\work\AAATeX.mtex

!!C:\work\AAATeX.txt

CopyFile["C:\work\AAATeX.txt", "C:\work\AAATeX.mtex"]

!!C:\work\AAATeX.mtex

TeXForm[MatrixForm[{{1, 2}, {3, 4}}]] >> C:\work\AAATeX.mtex

!!C:\work\AAATeX.mtex

Splice["C:\work\AAATeX.mtex"]

Splice["f:\Mathe/Daten/AAA.mtex"] (*win31...*)

!!C:\work\AAATeX.tex

!!f:\Mathe/Daten/AAA.tex (*win31...*)
```

■ Aufgabe 5 € Problème 5

Gegeben sei b[c, d]. Schreibe einen Befehl, der c als Subscript und d als Superscript wiedergibt!
• Soit donné b[c, d]. Ecris un ordre qui rend c comme subscript et d comme superscript.

Eine Lősung: • Une solution:

```
?Format

Format[b[c_,d_]] :=
  SequenceForm[Subscripted[b[c]],
  Superscript[d]];
b[c, d]

b[3, 5]
```

```
c[3, 5]
```

Was ist passiert?
• Que s'est-il passé?

■ Aufgabe 6 € Problème 6

Schreibe eine Funktion zur Berechnung der Glieder der Fibonacci-Folge bis zum n-ten Glied und speichere die Glieder sowie die Funktion in ein File.
• Ecris une fonction pour calculer les membres de la suite de Fibonacci jusqu'au n-ième membre et mémorise les membres ainsi que la fonction dans un fichier.

■ **Beispiel für das Programm**
€ Exemple pour le programme

```
Remove[fib];
fib[0]  = 1;
fib[1]  = 1;
fib[n_Integer]  = fib[n-1]+fib[n-2];
fib[n_]  = 0 ;
?fib

{fib[0],fib[1],fib[2],fib[3],fib[4],fib[5],
  "Test",fib[1.3]}

fib[10]
```

■ **Speichern der Funktion**
€ Mémoriser la fonction

```
Save["AAAfib", fib];
!!AAAfib

!!AAAfib

<<AAAfib

Save["f:\Mathe/Daten/AAA_fib", fib];
!!f:\Mathe/Daten/AAA_fib   (*win31...*)
```

■ **Erst rechnen und dann speichern**
€ D'abord calculer et ensuite mémoriser

```
Table[fib[n], {n, 0, 10}] >> C:\work\AAAfib

!!C:\work\AAAfib

Table[fib[n], {n, 0, 10}] >>
"f:\Mathe/Daten/AAA_fib";
!!f:\Mathe/Daten/AAA_fib (*win31...*)
```


■ **Ein Programm zum direkten Speichern der Glieder**
€ Un programme pour mémoriser directement les menbres

```
Remove[schreibFib];

schreibFib[fileName_String, zahl_Integer] :=
    Block[{ neufile = OpenWrite[fileName],
        n = 0},
        While[n<=zahl,
        WriteString[neufile,fib[n]];
        WriteString[neufile,"\t"];
        n++];
        (*WriteString[neufile,"\t"];*)
        Close[neufile]];

 schreibFib["f:\Mathe/Daten/AAA_fib", 12];
(*win31...*)

!!f:\Mathe/Daten/AAA_fib (*win31...*)

 schreibFib["C:\work\AAAfib", 12];

!!C:\work\AAAfib
```

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```

Kurs € Cours

15. Packages

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 15 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 15.
WIR94/99

Problem: Externe Programme und Programmpackete schreiben.
Mathematica selbst ist in C geschrieben. Externe Programme, sogenannte Packages, sind aber in Mathematica selbst geschrieben. Bsp.: "Calculus`DSolveIntegrals`"
• Problème: Ecris les programmes extérieurs et des paquets de programmes. *Mathematica* est écrit en C. Les programmes extérieurs, les dits Packages, sont écrits dans *Mathematica* même. Ex.: "Calculus`DSolveIntegrals`"

■ 15.1. Wieso Packages? Wie zugreifen?
€ Pourquoi des Packages? Comment les saisir?

■ 15.1.1. Wieso Packages?
€ Pourquoi des Packages?

Es ist klar, dass man häufig gebrauchte Funktionen oder Prozeduren etc. nicht immer wieder neu schreibt. Andere möchten auch gerne profitieren. So schreibt man eben externe Programme, die nach Bedarf zuladbar sind.
• Il est clair qu'on n'écrit pas chaque fois à nouveau des fonctions ou des procédures ets. fréquentes. Les autres aimeraient ausi en profiter. Ainsi on écrit des programmes extérieurs qu'on peut ajouter quand on en a besoin.

■ 15.1.2. Wie greife ich auf ein mitgeliefertes Package?
€ Comment saisit un Package livré

Die mitgelieferten Packages sieht man unter (Info, Open Function Browser, Packages) neu Help, Add-ons, Standard Packages, ... Nach dem Anklicken (Evaluate Template und dann Loaded Packages, Update Packages kann man im Browser das geladene Package anklicken und sieht die darin enthaltenen Funktionen) neu COPY-PASTE.
• Les Packages livrés se trouvent sous (Info, Open Function Browser, Packages) nouveau Help, Add-ons, Standard Packages, ... Après avoir marqué à la souris (Evaluate Template et ensuite Loaded Packages, Update Packages un peut marqué à la souris le Package chargé dans le Browser, on y voit les fonctions qui y xont contenues.) nouveau COPY-PASTE.

Lade das nachfolgende Package und schaue im Browser nach, welche Funktionen darin enthalten sind (Browser: Loaded Packages, Update Packages, anklicken etc.) neu Help, Add-ons, Standard Packages, ...
• Charge le Package suivant et regarde dans le Browser quelles fonctions y sont contenues. (Browser: Loaded Packages, Update Packages, marquer à la souris etc.) nouveau Help, Add-ons, Standard Packages, ...

```
Needs["Statistics`DescriptiveStatistics`"]
```

```
?Mean
```

■ 15.2. Contexts
€ Contexts

■ 15.2.1. Das Wesen des Contexts
€ L'essence du context

Ein Context ist vergleichbar mit einem Directory (Verzeichnis) oder einem Subdirectory (Folder). An die Namen der Variablen oder Symbolen wird dort der Contextname angehängt, so dass es sich dann um lokale Objekte handelt. Das verhindert Namenkollision. Contexte können geschachtelt sein. Bisher haben wir immer im globalen Context gearbeitet. Probiere aus:
• Un contexte est comparable à un directoire (index) ou à un sous-directoire (Subdirectory, Folder). On y ajoute le nom du contexte aux noms des variables on des symboles, de façon qu'il s'agisse ensuite d'objets locaux. Cela évite des collisions de noms. Les contexts peuvent être emvoîtés. Jusqu'à présent nous avons toujours travaillé dans le contexte global. Essaie:

```
?$Path
```

```
$Path
```

■ 15.2.2. Der Context-Pfad
€ Le chemin ("path") du contexte

Studiere die Befehle:
• Etudie les ordres:

```
?Context

?$Context

?$ContextPath

?*Context*

?Dir*

?Directory
```

Probiere aus: • Essaie:

```
$ContextPath

Context[ParametricPlot]

Context[Integrate]
```

■ 15.2.2. Context und Symbole
€ Contexte et symboles

■ Beispiele
€ Essaie

Studiere: • Etudie:

```
x = 2;
Remove[a];
{Context[x], Context[a], Context[Tuten]}
```

In welchem Context sind die geschaffenen Symbole?
• Dans quel contexte le trouvent les symboles crées?

```
Benzin = 50 Liter;
Auto`Benzin =
"Interessiert den Umweltschutz. --
Interessant pour la protection naturelle.";

?Benzin

?*`Benzin

?Auto`Benzin
```

Wird der Context "Global" angegeben bei einer Abfrage?
Teste das aus:
• Est-ce que le contexte "Global" est iniqué lors d'une interrogation? Fais un test:

```
`StudentError`Hoppla =
"Das war wohl nicht so gemeint!
Ce n'est certainement pas l'intention!";

`StudentError`Hoppla
```



```
Global`StudentError`Hoppla
```

■ 15.3. Context-Wechsel
€ Changement de contexte

■ 15.3.1. Allgemeines
€ Généralités

Dies ist möglich mit der Variable $Context. Studiere:
• Cela est possible avec la variable $Context. Etudie:

```
Benzin

$Context = "Auto`"

Benzin

$Context

$Context = "Global`"

Benzin
```

■ 15.3.2. Der Befehl "Begin"
€ L'ordre "Begin"

Mit den Befehlen Begin, BeginPackage, End, EndPackage wird der Context gewechselt. Studiere:
• Par les ordres Begin, BeginPackage, End, EndPackage on change de Context. Etudie:

```
?Begin

?BeginPackage

?End

?EndPackage
```

Probiere aus: • Essaie:

```
$Context

$ContextPath

Begin["Versuch`"]

$Context

$ContextPath
```

Nun führen wir eine Variable ein:
• Nous introduisons une variable:

```
dumm = "frech und beschränkt --
       mal poli et borné"

Context[dumm]

End[ ]

$Context

$ContextPath

?dumm

?Versuch`dumm
```

In welchem Context sind wir jetzt und wie ist der Pfad gesetzt?
• Dans quel contexte somme-nous maintenant et comment le chemin est-il posé?

■ 15.3.3. Der Befehl "Begin"
€ L'ordre "Begin"

Probiere aus: • Essaie:

```
BeginPackage["HalloWach`"]

$Context

$ContextPath
```

Was ist der Unterschied zwischen Begin und BeginPackage?
Probiere jetzt früher definierte Variablen aus:
• Auelle est la différence entre Begin et BeginPackage?
Essaie maintenant des variables définies il y a un certain temps:

```
?Benzin

?Global`Benzin

dumm

dumm =
"blöd und lieb zur Hälfte und beides
zusammen -- idiot et aimable à moitié
ou toutes les deux ensembles"

EndPackage[]

$Context

$ContextPath

dumm

Context[dumm]

{Versuch`dumm, dumm}
```


■ 15.4. Zum Laden von Packages
€ Pour charger des Packages

■ 15.4.1. Befehle
€ Ordres

Zur Verfügung stehen die Befehle <<, Get und Needs.
Studiere die Befehle:
• Les ordres suivants sont à disposition: <<, Get et Needs.
Etudie les ordres:

```
?<<

?Get

?Needs

?$Path

$Path
```

Hinweis: Die Variable $Path wird im File init.m gesetzt! (C:\Programme\Wolfram Research\Mathematica\3.0\Configuration\Kernel.) Dieses File kann von einem selbst modifiziert werden. (NeXT: Es soll sich bei privaten Installationen im eigenen Root befinden.) Beim Start von *Mathematica* wird es gelesen.
• Retenir: La variable $Path est mise dans le fichier init.m! (C:\Programme\Wolfram Research\Mathematica\3.0\Configuration\Kernel.) On peut modifier soi-même ce fichier. (NeXT: Il doit se trouver dans le propre Root quant il s'agit d'installations privées.) Il est lu quand on démarre *Mathematica*.

■ 15.4.2. Laden mit Get resp. <<
€ Charger avec Get resp. <<

Bemerkung: "Packages" befindet sich in $Path! Führe die folgenden Befehle nacheinander aus:
• Remarque: "Packages" se trouve dans $Path! Exécute les ordres suivants l'un après l'autre:

```
$Path

?Helix

Remove[Helix]

<< Graphics/Shapes.m;
?Helix

$ContextPath

<< Graphics`Shapes`;
?Helix

<< Graphics/Shapes.m;
?Helix
```

```
Needs["Graphics`Shapes`"]
```

```
$ContextPath
```

■ 15.4.3. Laden mit Needs
€ Charger avec Needs

Was ist der Unterschied gewesen zwischen Get und Needs?
• Quelle était la différence entre Get et Needs?

```
Get["Graphics`Shapes`"]
```

```
Needs["Graphics`Shapes`"]
```

Versuche: • Essaie:

```
BeginPackage["Graphics`Shapes`",
"Geometry`Rotations`"]
```

```
$ContextPath
```

■ 15.4.3. Wie sieht man den Inhalt eines Packages (Funktionen)?
€ Comment voit-on le contenu d'un Package (fonctions?)

Versuche: • Essaie:

```
?Rotate2D
```

```
?Geometry`Rotations`Rotate2D
```

```
?Graphics`Shapes`Helix (*Old Vers.*)
```

```
?Geometry`Rotations`*
```

```
?Graphics`Shapes`*     (*Old Vers.*)
```

```
?Geometry`*`*
```

Achtung: private bedeutet "zugriffgeschötzt"! Es wird keine spezifische Information abgegeben. Beispiel:
• Attention: Privé signifie "ne peut pas être saisi"! On ne donne pas d'information spécifique. Exemple:

```
?Geometry`Rotations`private`phi  (*Old Vers.*)
```

```
?Geometry`Rotations`private`*    (*Old Vers.*)
```

```
?Geometry`Rotations`private`vec  (*Old Vers.*)
```

Welche Packages sind da? Wir suchen z.B. ein Objekt, in dessen Namen hinter dem "`" ein "a" vorkommt:
• Quels Packages y a-t-il? Nous cherchons p.ex. un objet, dans le nom duquel il y a un "`" suivi d'un "a" :

```
?*`a*
```


■ 15.5. Der Package-Stil
€ Le style Package

Studiere die folgenden Beispiele konsequent durch!
• Fais une étudie approfondie des exemples suivants!

■ **Hier zwei Beispiele aus der *Mathematica*-Bibliothek**
€ Voici deux exemples de la bibliothèque de *Mathematica*:

■ **Beispiel 1 (Kopie aus der old Library, etwas kommentiert)**
€ Exemple 1 (Copie de la old Librery, un peu commentée)

```
(*************************************************************************

    Adapted from
    Roman E. Maeder: Programming in Mathematica,
    Second Edition, Addison-Wesley, 1991.
Wichtig: Gute Dokumentation! •  Important: Bonne documentation!

 *************************************************************************)


BeginPackage["Graphics`ArgColors`"]
(* Wichtig: $Context und $ContextPath wechseln!                    *)
(*• Important: Changer $Context et $ContextPath !                  *)
(* Wichtig: Die hier definierten Funktionen, z.B. ArgColor, sind   *)
(*        nach aussen sichtbar. Die Symbole daför werden in diesem *)
(*        Context kreiert.                                         *)
(* • Important: Les fonctions définis ici, p.ex. ArgColor, sont    *)
(*        visibles à l'extérieur. Les smaboles en sont crées dans ce *)
(*        contexte.                                                *)

ArgColor::usage = "ArgColor[z] gives a color value whose hue is proportional
      to the argument of the complex number z."
ArgShade::usage = "ArgShade[z] gives a gray level proportional
      to the argument of the complex number z."

ColorCircle::usage = "ColorCircle[r, (light:1)] gives a color value whose hue
      is proportional to r (mod 2Pi) with lightness light."
(* Wichtig: usage-Statement für alle aussen sichtbaren Funktionen *)
(* • Important: usage-Statement visibles pour toutes les fonctions *)
(*    visibles à l'extérieur. *)

Begin["`Private`"]
(* Wichtig: $Context wechselt auf `Private`,                       *)
(* • Important: $Context devient `Private`,                        *)
(*        $ContextPath bleibt • reste.  *)
(* Wichtig: Hier neu definierte Funktionen sind aussen unsichtbar! *)
(* • Important: Les fonctions définis ici nouvellement ne sont pas*)
(*    visibles à l'extérieur! *)

HelperFunction[z_] := Print["Mich siehst Du von assen nicht", z]
```



```
ArgColor[z_] := RGBColor[1, 1, 1] /; z == 0.0

ArgColor[z_] := ColorCircle[ Arg[z] ] /; NumberQ[N[arg]]

ArgShade[z_] := GrayLevel[1] /; z == 0.0

ArgShade[z_] := GrayLevel[N[(Pi + Arg[z])/(2Pi)] ] /; NumberQ[N[Arg[z]]]

ColorCircle[arg_, light_:1.0] := Hue[ N[arg/2/Pi], 1, light] /; NumberQ[N[arg]]

End[ ]
(* Wichtig: $Context wechselt zurück                               *)
(*        auf den Wert vor dem Aufruf von `Private`                *)
(* •  Important: $Context reprend la forme d'origine               *)
(*        sur la valeur d'avant l'apppel de  `Private`             *)

Protect[ArgColor, ArgShade, ColorCircle]
(* Wichtig: Die aussen sichtbaren Funktionen werden geschützt.     *)
(* •  Important: Les fonctions visibles à l'extérieur sont protégées.  *)

EndPackage[ ]
(* Wichtig: $Context wird zurückgewechselt auf den alten Wert.     *)
(* Wichtig: Zu $ContextPath wird der Name                          *)
(*        "Graphics`ArgColors`" hinzugefügt.                       *)
(* •  Important: $Context reprend la valeur d'origine.             *)
(* •  Important: On ajoute à $ContextPath le nom                   *)
(*        "Graphics`ArgColors`" .                                  *)
```

```
(*********************************************************************

Adapted from
Roman E. Maeder: Programming in Mathematica,
Second Edition, Addison-Wesley, 1991.
(Obige Kopie aus der Library etwas abgeändert,
 so dass sie auf NeXT... lauffähig ist.
 € La copie de Libary ci cessus un peu changée,
  de façon qu'elle puisse fonctionner sur NeXT
  .....)

 *********************************************************************)

BeginPackage["Test`"];

argColor1::usage =
"ArgColor[z] gives a color value whose hue is
    proportional;
    to the argument of the complex number z."
argShade1::usage =
"ArgShade[z] gives a gray level proportional
    to the argument of the complex number z.";

colorCircle1::usage =
"ColorCircle[r, (light:1)] gives a color value
     whose hue is proportional to r (mod 2Pi)
     with lightness light.";

Begin["`Private`"];

argColor1[z_] := RGBColor[1, 1, 1] /;
                 z == 0.0  ;

argColor1[z_] := ColorCircle[ Arg[z] ] /;
                 NumberQ[N[Arg]]  ;

argShade1[z_] := GrayLevel[1] /; z == 0.0
argShade1[z_] := GrayLevel[N[(Pi +
        Arg[z])/(2Pi)] ] /;
        NumberQ[N[Arg[z]]]   ;

colorCircle1[arg_, light_:1.0] :=
        Hue[ N[arg/2/Pi], 1, light] /;
        NumberQ[N[Arg]]   ;

End[];

Protect[argColor1, argShade1, colorCircle1]

EndPackage[];


(*********************************************************************

Adapted from
Roman E. Maeder: Programming in Mathematica,
Second Edition, Addison-Wesley, 1991.
(Obige Kopie aus der Library etwas abgeändert,
 so dass sie auf NeXT.. lauffähig ist, jetzt mit
 Abfrage der Context-Variablen und des Context-
 Pfades.
 € La copie de Libary ci cessus un peu changée,
```



```
  de façon qu'elle puisse fonctionner sur NeXT
  .., maintenant avec interrogation (appel) des
  variables du contexte et du chemin du
  contexte.)

 *********************************************************************)

Print["1. Abfrage: $ContextPath = ",
    $ContextPath, " $Context = ", $Context];

BeginPackage["Test`"];
Print["2. Abfrage: $ContextPath = ",
    $ContextPath, " $Context = ", $Context];

argColor1::usage =
"ArgColor[z] gives a color value whose hue is
    proportional;
    to the argument of the complex number z."
argShade1::usage =
"ArgShade[z] gives a gray level proportional
    to the argument of the complex number z.";

colorCircle1::usage = "ColorCircle[r,
    (light:1)] gives a color value
     whose hue is proportional to r (mod 2Pi)
     with lightness light.";

Begin["`Private`"];
Print["3. Abfrage: $ContextPath = ",
    $ContextPath, " $Context = ", $Context];

argColor1[z_] := RGBColor[1, 1, 1] /;
        z == 0.0  ;

argColor1[z_] := ColorCircle[ Arg[z] ] /;
        NumberQ[N[Arg]]  ;

argShade1[z_] := GrayLevel[1] /;
        z == 0.0
argShade1[z_] := GrayLevel[N[(Pi +
        Arg[z])/(2Pi)] ] /;
        NumberQ[N[Arg[z]]]   ;

colorCircle1[arg_, light_:1.0] :=
        Hue[ N[arg/2/Pi], 1, light] /;
        NumberQ[N[Arg]]   ;

End[];
Print["4. Abfrage: $ContextPath = ",
    $ContextPath, " $Context = ", $Context];

Protect[argColor1, argShade1, colorCircle1]

EndPackage[];
Print["5. Abfrage: $ContextPath = ",
    $ContextPath, " $Context = ", $Context];
```

- **Beispiel 2 (Kopie aus der Library, ohne Kommentar)**
  € Exemple 2 (Copie de Library, sans commentaire)

```
(*************************************************************************

        Copyright 1991 by Roman E. Maeder

        Adapted from
        Roman E. Maeder: Programming in Mathematica,
        Second Edition, Addison-Wesley, 1991.

        Permission is hereby granted to make copies of this file for
        any purpose other than direct profit, or as part of a
        commercial product, provided this copyright notice is left
        intact.  Sale, other than for the cost of media, is prohibited.

        Permission is hereby granted to reproduce part or all of
        this file, provided that the source is acknowledged.

 *************************************************************************)

        (* Enclosed with Mathematica by permission *)


BeginPackage["Examples`Collatz`"]

Collatz::usage =
        "Collatz[n] gives a list of the iterates in the 3n+1 problem,
        starting from n. The conjecture is that this sequence always
        terminates."

StoppingTime::usage = "StoppingTime[n] finds the total stopping time
        of the integer n. This is the length of the Collatz sequence
        before hitting 1."

FindMaxima::usage = "FindMaxima[from] reports successive maxima of the
        total stopping time starting the search with the integer from."

Begin["`Private`"]

Collatz[n_Integer] := AppendCollatz[{}, n]

AppendCollatz[sofar_, 1] := Flatten[{sofar, 1}]

AppendCollatz[sofar_, n_Integer] :=
        AppendCollatz[{sofar, n}, 3 n + 1]     /; OddQ[n]

AppendCollatz[sofar_, n_Integer] :=
        AppendCollatz[{sofar, n}, n / 2]       /; EvenQ[n] && n != 0

StoppingTime[n_Integer?Positive] :=
```



```
        Module[{i=1, m=n}, While[m != 1, m = If[OddQ[m], 3m+1, m/2]; i++];i ]

FindMaxima[low_] :=
        Module[{m=0, n=0, i=low, j},
                While[ True,
                        j = StoppingTime[i];
                        If[j > m, m = j; n = i;
                                Print["StoppingTime[", n, "] = ", m] ];
                        i++;
                        If[ Mod[i, 100] == 0, Print["i = ", i] ]
                ]
        ]

End[]

Protect[ Collatz, StoppingTime, FindMaxima ]

EndPackage[ ]
```

■ 15.6. Namenskonflikte
€ Conflicts de noms

■ 15.6.1. Beispiel eines Konflikts
€ Exemple d'un conflict

■ Mehrfach verwendete Namen in verschiedenen Umgebungen
€ Noms utilisés de plusieurs façons dans des environs différents.

Studiere die folgende Abfolge:
• Etudier la suite suivante:

```
LaplaceTransform[(1+4t+5t^2) Exp[-3t], t, s]
```

Die gerufene Funktion war noch unbekannt. Das Symbol LaplaceTransform ist aber jetzt kreiert worden, allerdings ohne konkreten Inhalt.
• La fonction appelée était encore inconnue. Le symbole LaplaceTransform a été crée maintenant, cependant sans contenu concret.

```
?LaplaceTransform
```

Laden des Packages:
• Charger le Package:

```
Needs["Calculus`LaplaceTransform`"]

?LaplaceTransform
```

Die in Global` vorher kreierte Funktion überschattet immer noch die in einem "Unter-Context" zugeladene Funktion mit demselben Namen.
• La fonction créée auparavant dans Global` recouvre encore la fonction du même nom qui se trouve dans un sous-contexte.

```
Remove[LaplaceTransform]
```

Die in Global` existierende Funktion wird eliminiert. Die darunter liegende zeigt sich jetzt.
• La fonction existant dans un Global` est éliminée. Celle qui se trouve dessous devient visible.

```
?LaplaceTransform
```

Nun kann man sie auch verwenden:
• Maintenant on peut aussi l'utiliser:

```
LaplaceTransform[(1+4t+5t^2) Exp[-3t], t, s]
```

Was gibt es sonst noch im geladenen Package?
• Qu'y a-t-il encore dans le Package chargé?

```
?Calculus`LaplaceTransform`*
```

■ 15.6.2. Der empfohlene Stil
€ Le style conseillé

Im Folgenden wird ein Prototyp eines Packages demonstriert, das ursprönglich von Henry Cejtin und Theo Gray stammen soll:
• Dans ce qui suit on démontre un prototype d'un Package qui a été créé, soit-disant, par Henry Cejtin et Theo Gray.

```
(* Old - for NeXT etc. *)
OpenWrite["f:\mathe/Daten/AAKurs15.m"];
WriteString["f:\mathe/Daten/AAKurs15.m", "\n",
"Du sollst das Package von Hand in das File",
"\n", "AAKurs15.m kopieren."
];
(* End WriteString  etc. - continued .... *)

" " >> AAKurs15.m;

OpenWrite[
"AAKurs15.m"];
WriteString[
"AAKurs15.m",
 "\n",
"!!!!!!
Du sollst das Package von Hand in das File",
"\n","AAKurs15.m kopieren."
"Il faut copier le Package à la main dans le
 File", "\n","AAKurs15.m kopieren." ];
(* End WriteString   *)

 Close["AAKurs15.m"];


(*
```



```
 * AAKurs15.m
 *
 * This package demonstrates the recommended
 * package style.
 *
 *)

(*
 * Prepend the context ProtoNotebook` to
 * $ContextPath.
 * Then Needs will be able to determine when
 * the package
 * has been loaded.
 *)
BeginPackage["AAKurs15`"];
EndPackage[];

(*
 * Now the usage statements.
 *)
ExamplePlot::usage = "ExamplePlot[n,m] draws \
    n starbursts and m spirals. \
    ExamplePlot[n] draws n starbursts and
    no spirals.";

SamplePlot::usage =
    "SamplePlot is Vaporware. It \
    does nothing.";

(*
 * Now the global objects.
 *)
ExamplePlot;
SamplePlot;

(*
 * Now the private context. The objects here
 * are not \
 * intended to be accessed by the user.
 *)
 Begin["AAKurs15`Private`"];

(*
 * Local objects.
 *)
`starBurst;
`spiral;

(*
 * Define starBurst and spiral.
 *)
 starBurst[center_, radius_, n_] :=
    Block[
        { `r  },
        Table[
            Line[{center,
                  center + radius {Cos[r],
                  Sin[r]}
                }],
                {r, 0, 2Pi - 2Pi/n, 2Pi/n}
            ]
          ];
```

```
spiral[center_, radius_, loops_] :=
    Block[
        { `t  },
        Line[
            Table[center +
                  radius t {Cos[t], Sin[t]}/(2Pi
                                    loops),
                  {t, 0, 2Pi loops, 2Pi/30}
                 ]
              ]
          ];

ExamplePlot[stars_] :=
    Show[
        Graphics[
            Table[
                starBurst[
                    4 {Random[], Random[]},
                    0.3 + Random[],
                    20
                         ],
                         {stars}
                  ]
                ],
                AspectRatio -> Automatic
          ];

ExamplePlot[stars_, spins_] :=
    Show[
        Graphics[{
            Table[
                starBurst[
                    4 {Random[], Random[]},
                    0.3 + Random[],
                    20
                         ],
                         {stars}
                  ],
             Table[
                spiral[
                    4 {Random[], Random[]},
                    0.3 + Random[],
                    Random[Integer, {2, 5}]
                         ],
                         {spins}
                   ]
                }],
                AspectRatio -> Automatic
          ];

(*
 * Protect now the function and lock the private
 * functions.
 *)
 Protect[ExamplePlot, SamplePlot];
 Append[Attributes[ExamplePlot], Locked];
 Append[Attributes[SamplePlot], Locked];

 End[];
```


Schaue ins File • Regarde le File
C:\Programme\Wolfram Research\Mathematica\3.0\AAKurs15.nb !!!!!!!!!!!!

```
Needs["AAKurs15`"]

<< AAKurs15

?AA*

$Path

Needs["AAKurs15`"]

?SamplePlot

?ExamplePlot

ExamplePlot[4, 3]

ExamplePlot[5]

SamplePlot[5]
```

■ 15.6.3. Einlesen eines externen Files in einen andern Context
€ Lire un fichier extérieur et le mettre dans un autre contexte

Im folgenden Beispiel wird gezeigt, wie ProtoNotbook` in den Context PlottingExamples` eingelesen wird. Probiere die verschiedenen Situationen durch:
• Dans l'exemple suivant on montre comme ProtoNotbook` est mis dans
le contexte PlottingExamples`. Essaie les différentes situations:

```
?SymbolicSum

Remove["Global`@*"]

End[]

?*SamplePlot

$Context

Begin["PlottingExamples`"];
$Context

$ContextPath

?SymbolicSum

?*SamplePlot

Needs["Algebra`SymbolicSum`"]

?Algebra`SymbolicSum`*

$Context

$ContextPath
```

```
a = 1

a

End[]

a

$Context

$ContextPath

?Algebra`SymbolicSum`*
```

■ 15.6.4. Notebooks
€ Notebooks

Dieses File hier ist ein Notebook. In solchen Notebooks oder allgemein bei der interaktiven Benützung von Mathematica mit einem Front-End, das mit Zellen arbeitet, können Probleme mit dem Context auftreten, wenn man nicht aufpasst. Studiere das folgende Beispiel:

• Le fichier est un Notebook. Dand des Notebooks de ce genre ou plus généralement quand on utilise ineractivement *Mathematica* avec un Frnt-End qui travaille avec des cellules, il peut y avoir des problèmes avec le contexte quand on ne fait pas attention. Etudie l'exemple suivant:

```
Remove["Global`@*"]

esel = "Dummes menschliches Individuum,
Anwesende immer ausgeschlosen. --- Etre humain
bête, personnes présents exclus."

Context[esel]

BeginPackage["Zoologie"];
esel = "Säugetier im Dienste des Menschen.
       --- Mammifèr au service des hommes";

esel

Context[esel]

Global`esel

Zoologie`esel

EndPackage[]

esel
```

Wo ist der eingangs eingegebene "Wert" der Variablen "esel" geblieben?

• Où est disparu la "valeur" de la variable "esel" entrée au début?

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```


Uebungen € Exercices

15. Packages

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach....
Hinweis: Kapitel 15 lesen!
€ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 15.
WIR94/99

■ Aufgabe 1 € Problème 1

Lade das Package "Colors.m" aus "Graphics". Liste alle Farben auf.

• Charger le Package "Colors.m" de "Graphics". Fais une liste des couleurs.

```
Needs["Graphics`Colors`"]

?Graphics`Colors`*
```

■ Aufgabe 2 € Problème 2

Schreibe ein *Mathematica*-Package, das den Mittelwert (mean) und den Median einer Liste berechnet. In der folgenden Lösung wird die Standard-Lösung aus den mitgelieferten Packages kopiert und präsentiert.

• Ecris un Package de *Mathematica*, qui calcule la valeur moyenne et la médiane d'une liste. Dans la solution suivante, la solution standard est copiée et présentée dans les Packages livrées.

Standard-Package einlesen:

• SLre le Package standard:

```
Needs["Statistics`DescriptiveStatistics`"]

?Median

?Mean
```

Standard-Package anschauen:

• Regarde le Package standard:

```
!!Statistics`DescriptiveStatistics`

?ReadList
```

```
Options[Put]

?NullRecords

readIn = ReadList[
    "Statistics`DescriptiveStatistics`",
        Record,RecordSeparators -> {"\n","\t"}];
readIn >> f:\Mathe/Daten/DataRec;
readIn >> f:\Mathe/Daten/DataRec.ma;
readIn = ReadList[
    "Statistics`DescriptiveStatistics`",
                    String];
readIn >> f:\Mathe/Daten/DataOutS;
readIn >> f:\Mathe/Daten/DataOutS.ma;
```

Diese Files können nun editiert und von Hand manipuliert werden. Z.B. wie folgt:

- Ces fichiers peuvent être édités maintenant et manipulés à la main. P. ex. de la façon suivante:


```
OpenWrite["Statisti.m"];
WriteString["Statisti.m",
"\n",
"Du sollst das Package von Hand in das File","\n",
"Statisti.m kopieren."
];   (* End WriteString   *)

Close["Statisti.m"];


(* Author:  Student Heimlich Fileklau  *)
(* History: Original version ... 1989. *)
(*Revised 1995.                        *)
(* Summary: This package computes descriptive *)
(*                          statistics *)
(* Keywords: mean, median *)

BeginPackage["AAAStatistics`"];
EndPackage[ ];

mean::usage =
    "mean[list] gives the mean of the entries \
                        in list.";
median::usage =
    "median[list] gives the median of the  \
                  entries in list.";

Begin["`Private`"];
mean[list_] := Apply[Plus, list]/Length[list] /;
               VectorQ[list] && Length[list] > 0;
median[list_] := Sort[list][[
                        (Length[list]+1)/2]] /;
               VectorQ[list] &&
                        OddQ[Length[list]];
median[list_] := Block[{s, n},
           s =Sort[list] ;
           n = Length[list] ;
           (s[[n/2]] + s[[n/2 + 1]]) / 2] /;
           VectorQ[list] && EvenQ[Length[list]];
End[ ];

(* Protect[mean, median];
SetAttributes[ mean, ReadProtected];
SetAttributes[ median,ReadProtected];  *)


Needs["AAAStatistics`"]

?AAAStatistics`*

?mean

?median

mean[{1,2,3,4,5,6,7,8,9}]

mean[{1,2,3,4,5,6,7,8}]

median[{1,2,3,4,5,6,7,8}]

median[{1,2,3,4,5,6,7,8,9}]
```

Zum Beispielprogramm: Funktioniert es immer gut und somit tadellos? - Wenn nicht, so korrigiere es! Wenn Du Erfolg hast und es besser machen kannst, so hast Du ein grosses Ziel erreicht!
• Quant au programme exemple: Fonctione-t-il toujours bien et donc sans erreurs? - Si non, corrige-le. Si tu as succès et que tu peux le faire mieux, tu as atteint un grand but!

■ "Putzmaschine" einsetzen
€ Employer la "machine de nettoyage"

```
Remove["Global`@*"]
```