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!
Run mit WIN+*Mathematica* Version 5.2

▮ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 7.
Testé avec *Mathematica* version 5.2+WIN

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:

```
In[1]:= a = 5

Out[1]= 5
```

In[2]:= **a + 4**

Out[2]= 9

Statt Werte kann man auch Ausdrücke zuordnen und damit rechnen:
■ Au lieu des valeurs on peut aussi assigner des expressions et calculer avec celles-ci:

In[3]:= **a = 3x + y**

Out[3]= 3 x + y

In[4]:= **1/a**

Out[4]= $\frac{1}{3\,x + y}$

"." heisst "missing value". Weist man "." zu, so ist die Variable "leer":
■ "." veut dire "missing value". Si on assigne ".", la variable est "vide":

In[5]:= **a = .**

In[6]:= **a**

Out[6]= 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:

In[7]:= **?Set**

```
lhs = rhs evaluates rhs and assigns the result to be the value of lhs. From then on, lhs is
   replaced by rhs whenever it appears. {l1, l2, ... } = {r1, r2, ... } evaluates
   the ri, and assigns the results to be the values of the corresponding li. Mehr...
```

In[8]:= **?=**

```
lhs = rhs evaluates rhs and assigns the result to be the value of lhs. From then on, lhs is
   replaced by rhs whenever it appears. {l1, l2, ... } = {r1, r2, ... } evaluates
   the ri, and assigns the results to be the values of the corresponding li. Mehr...
```

In[9]:= **Set[a, 33444]**

Out[9]= 33444

In[10]:= **a**

Out[10]= 33444

In[11]:= **random1 = Random[]**

Out[11]= 0.529863

In[12]:= **2**

Out[12]= 2

In[13]:= **3 + random1**

Out[13]= 3.52986

":=" 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:

In[14]:= **?:=**

```
lhs := rhs assigns rhs to be the delayed value of lhs. rhs is maintained in an unevaluated
  form. When lhs appears, it is replaced by rhs, evaluated afresh each time. Mehr…
```

In[15]:= **?SetDelayed**

```
lhs := rhs assigns rhs to be the delayed value of lhs. rhs is maintained in an unevaluated
  form. When lhs appears, it is replaced by rhs, evaluated afresh each time. Mehr…
```

In[16]:= **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:

In[17]:= **Table[random1,{5}]**

Out[17]= {0.529863, 0.529863, 0.529863, 0.529863, 0.529863}

In[18]:= **Table[random2,{5}]**

Out[18]= {0.320157, 0.260433, 0.706004, 0.425192, 0.767409}

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 pourquoi apparaît-elle? (Tu peux essaier plusieurs fois.) Application à la définition de fonctions:

In[19]:= **Remove[f];Remove[g];**

a) Unmittelbare Zuweisung. Beobachte, was b bewirkt:
■ Application immédiate. Observe ce que produit b:

In[20]:=
```
b = 3;
f[x_] = x + b
```

Out[21]= 3 + x

Veränderung des Wertes und Aufruf: Was bewirkt b?
■ Changement des valeurs et appel: Que produit b?

In[22]:=
```
b = 100;
f[5]
```

Out[23]= 8

b) Verzögerte Zuweisung. Beobachte, was b bewirkt:
■ Application retardée. Observe ce que produit b:

```
In[24]:= 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?

```
In[26]:= b = 100;
         g[5]

Out[27]= 105
```

Abruf der Definitionen:

■ Appel des définitions:

```
In[28]:= ?f

         Global`f

         f[x_] = 3 + x

In[29]:= ?g

         Global`g

         g[x_] := x + b
```

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:

```
In[30]:= a1 = a2 = a3

Out[30]= a3
```

Oder listenweise:

■ Ou par listes:

```
In[31]:= {u, v} = {4, 15}

Out[31]= {4, 15}

In[32]:= {s, t} = {u, v}

Out[32]= {4, 15}

In[33]:= {t, s}

Out[33]= {15, 4}
```

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)

In[34]:= **??Out**

```
%n or Out[n] is a global object that is assigned to be the value produced
   on the nth output line. % gives the last result generated. %% gives the
   result before last. %% ... % (k times) gives the kth previous result. Mehr...

Attributes[Out] = {Listable, Protected}

Out[0] = 0

%1 = 5

%2 = 9

%3 = 3 x + y

%4 = 1/(3 x+y)

%5 = Null

%6 = a

%7 = {Null}

%8 = {Null}

%9 = 33444

%10 = 33444

%11 = 0.529863

%12 = 2

%13 = 3.52986

%14 = {Null}

%15 = {Null}

%16 = Null

%17 = {0.529863, 0.529863, 0.529863, 0.529863, 0.529863}

%18 = {0.320157, 0.260433, 0.706004, 0.425192, 0.767409}

%19 = Null

%20 = 3

%21 = 3 + x

%22 = 100

%23 = 8

%24 = 3

%25 = Null

%26 = 100

%27 = 105
```

```
%28 = {Null}

%29 = {Null}

%30 = a3

%31 = {4, 15}

%32 = {4, 15}

%33 = {15, 4}
```

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 factoriels:

```
In[35]:= Remove[fac];
         fac[1]  = 1;
         fac[n_]:= fac[n] = n fac[n-1];

In[38]:= ?fac

         Global`fac

         fac[1] = 1

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

In[39]:= Table[fac[n],{n,1,5}]

Out[39]= {1, 2, 6, 24, 120}

In[40]:= fac[50];
         Timing[fac[50]]

Out[41]= {0. Second, 30414093201713378043612608166064768844377641568960512000000000000}
```

Probiere: ▮ Essaie:

```
In[42]:= Remove[fak];
         fak[1]  = 1;
         fak[n_]:= n fak[n-1];
         Table[fak[n],{n,1,5}]

Out[45]= {1, 2, 6, 24, 120}

In[46]:= fak[50];
         Timing[fak[50]]

Out[47]= {0. Second, 30414093201713378043612608166064768844377641568960512000000000000}
```

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:

```
In[48]:= Remove[x] ; x = 1 ; Print["1)   ", x] ;
         x = . ; Print["2)   ",x]

         1)   1

         2)   x

In[50]:= ??x

         Global`x

In[51]:= x = 1 ; Print["1)   ", x]

         1)   1

In[52]:= ??x

         Global`x

         x = 1

In[53]:= x = . ; Print["2)   ",x]

         2)   x

In[54]:= ??x

         Global`x
```

7.2.2. Funktion löschen:
▮ Effacer la fonction

```
In[55]:= ?fak

         Global`fak

         fak[1] = 1

         fak[n_] := n fak[n - 1]

In[56]:= fak[1] = .

In[57]:= fak[n_] = .
```

In[58]:= **?fak**

```
Global`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:

In[59]:= **kubus[x_] = x^3**

Out[59]= x^3

In[60]:= **?kubus**

```
Global`kubus

kubus[x_] = x^3
```

In[61]:= **Clear[kubus]**

In[62]:= **?kubus**

```
Global`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 connu.

In[63]:= **Attributes[kubus] = {Listable}**

Out[63]= {Listable}

In[64]:= **Clear[kubus]**

In[65]:= **?kubus**

```
Global`kubus

Attributes[kubus] = {Listable}
```

Sogar das Attribut ist noch bekannt.

- Même l'attribut est encore connu.

In[66]:= **ClearAll[kubus]**

In[67]:= **?kubus**

```
Global`kubus
```

Jetzt ist das Attribut nicht mehr bekannt.

- Maintenant l'attribut n'est plus connu.

In[68]:= **Remove[kubus]**

In[69]:= **?kubus**

```
Information::notfound : Symbol kubus not found. Mehr…
```

Jetzt ist auch der Name nicht mehr bekannt.
■ Maintenant même le nom n'est plus connu.

In[70]:= **Remove[Apply]**

```
Remove::rmptc : Symbol Apply is Protected and cannot be removed. Mehr…
```

(Hier handelt es sich um einen *Mathematica*-Namen!)
■ (Ici il s'agit d'un nom de *Mathematica*))

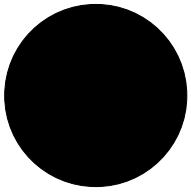
7.2.3. Probleme mit Packages:
■ Problèmes de Packages:

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 l'appel du package. Mais comme la fonction a déjà été enregistrée au premier appel, elle ne se laisse pas surcharger maintenant. Il faut d'abord l'effacer. La fonction chargée du Package reste. Exemple:

In[71]:=
```
Show[Graphics[{CMYColor[0.6,0.5,0.4],PointSize[0.3],
          Point[{0,0}]}]];
```

```
General::spell1 :
 Possible spelling error: new symbol name "CMYColor" is similar to existing symbol "CMYKColor". Mehr…

Graphics::gprim :
 CMYColor[0.6, 0.5, 0.4] was encountered where a Graphics primitive or directive was expected. Mehr…
```



In[72]:= **Needs["Graphics`Colors`"]**

```
CMYColor::shdw : Symbol CMYColor appears in multiple contexts {Graphics`Colors`, Global`};
   definitions in context Graphics`Colors` may shadow or be shadowed by other definitions. Mehr…
```

In[73]:= **?CMYColor**

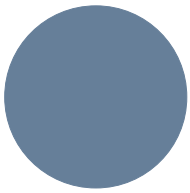
```
Global`CMYColor
```

In[74]:= **Remove[CMYColor]**

In[75]:= **?CMYColor**

```
CMYColor[c,m,y] represents a color in the CMY (cyan-magenta-yellow) system. Mehr…
```

```
In[76]:= Show[Graphics[{CMYColor[0.6, 0.5, 0.4], PointSize[0.3], Point[{0, 0}]}]];
```



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".

```
In[77]:= ?fak

         Global`fak
```

```
In[78]:= Remove["Global`*"]
```

```
In[79]:= ?fak

         Information::notfound : Symbol fak not found. Mehr…
```

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 "protected".

```
In[80]:= (*   Remove["@*"] old, Remove["Global`*"] new   *)
```

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

```
In[81]:= ?/.

         expr /. rules applies a rule or list of rules in
            an attempt to transform each subpart of an expression expr. Mehr…

In[82]:= ?ReplaceAll

         expr /. rules applies a rule or list of rules in
            an attempt to transform each subpart of an expression expr. Mehr…

In[83]:= ?Replace

         Replace[expr, rules] applies a rule or list of rules in an
            attempt to transform the entire expression expr. Replace[expr, rules,
            levelspec] applies rules to parts of expr specified by levelspec. Mehr…
```

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:

```
In[84]:= x + 7y

Out[84]= x + 7 y

In[85]:= x + 7y /. x->4

Out[85]= 4 + 7 y

In[86]:= x + 7y /. {x->4, y->3}

Out[86]= 25

In[87]:= x + 7y

Out[87]= x + 7 y
```

Die Ersetzung ist also nicht bleibend! Beachte:

∎ Le remplacement n'est pas définitif! Observe:

```
In[88]:= 3x + 7x /. {3x->y, y->3}

Out[88]= 10 x
```

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:

```
In[89]:= ?->
```

```
lhs -> rhs represents a rule that transforms lhs to rhs. Mehr…
```

```
In[90]:= ?:>
```

```
lhs :> rhs represents a rule that transforms
   lhs to rhs, evaluating rhs only when the rule is used. Mehr…
```

Man hat also eine ähnliche Situation wie bei "=" und ":=". Beispiele:
▮ On a donc une situation semblable comme pour "=" et ":=". Exemple:

```
In[91]:= Table[x, {6}]
Out[91]= {x, x, x, x, x, x}

In[92]:= Table[x, {6}] /. x-> Random[]
Out[92]= {0.357011, 0.357011, 0.357011, 0.357011, 0.357011, 0.357011}
```

Offenbar ist erst gerechnet worden. Dann wurde Random[ ] einmal ausgeführt, darauf zugewiesen. x bleibt leer:
▮ Apparemment on a d'abord calculé. Ensuite on a exécuté une fois Random[ ], puis assigné. x reste vide.

```
In[93]:= x
Out[93]= x

In[94]:= Table[x, {6}] /. x:> Random[]
Out[94]= {0.799552, 0.201374, 0.507658, 0.690089, 0.183678, 0.286869}
```

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:

```
In[95]:= ?//.
```

```
expr //. rules repeatedly performs replacements until expr no longer changes. Mehr…
```

Vergleiche: ▮ Compare:

In[96]:= **log[a b c d] /. log[x_ y_] -> log[x] + log[y]**

```
General::spell1 :
 Possible spelling error: new symbol name "log" is similar to existing symbol "Log". Mehr…
```

Out[96]= log[a] + log[b c d]

In[97]:= **log[a b c d] //. log[x_ y_] -> log[x] + log[y]**

Out[97]= log[a] + log[b] + log[c] + log[d]

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:

In[98]:= **Options[ContourPlot]**

Out[98]=
```
{AspectRatio → 1, Axes → False, AxesLabel → None, AxesOrigin → Automatic,
 AxesStyle → Automatic, Background → Automatic, ColorFunction → Automatic,
 ColorFunctionScaling → True, ColorOutput → Automatic, Compiled → True,
 ContourLines → True, Contours → 10, ContourShading → True,
 ContourSmoothing → True, ContourStyle → Automatic, DefaultColor → Automatic,
 DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction,
 Epilog → {}, FormatType :→ $FormatType, Frame → True, FrameLabel → None,
 FrameStyle → Automatic, FrameTicks → Automatic, ImageSize → Automatic,
 PlotLabel → None, PlotPoints → 25, PlotRange → Automatic, PlotRegion → Automatic,
 Prolog → {}, RotateLabel → True, TextStyle :→ $TextStyle, Ticks → Automatic}
```

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:

In[99]:= **?$Line**

```
$Line is a global variable that specifies the number of the current input line. Mehr…
```

In[100]:=
$Line

Out[100]=
100

In[101]:=
?InString

```
InString[n] is a global object that is assigned to be the text of the nth input line. Mehr…
```

In[102]:=

```
?ToString
```

```
ToString[expr] gives a string corresponding to
   the printed form of expr in OutputForm. ToString[expr, form] gives
   the string corresponding to output in the specified form. Mehr...
```

In[103]:=

```
?StringJoin
```

```
"s1" <> "s2" <> ... , StringJoin["s1", "s2", ... ] or StringJoin[
   {"s1", "s2", ... }] yields a string consisting of a concatenation of the si. Mehr...
```

In[104]:=

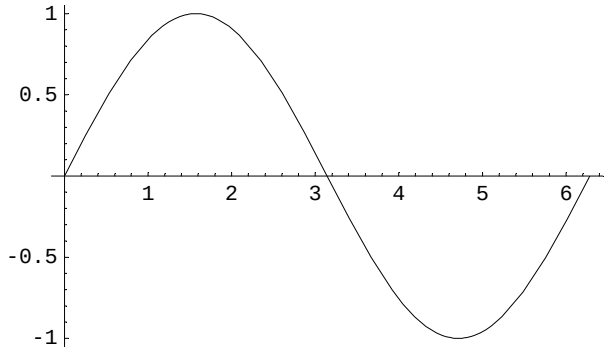
```
?SetOptions
```

```
SetOptions[s, name1->value1, name2->value2, ... ] sets the specified default
   options for a symbol s. SetOptions[stream, ... ] or SetOptions["name", ... ]
   sets options associated with a particular stream. SetOptions[object, ... ] sets
   options associated with an external object such as a NotebookObject. Mehr...
```

In[105]:=

```
SetOptions[Plot,PlotLabel :> StringJoin[
            "In[", ToString[$Line], "] :> ",
            InString[$Line]
            ]];
Plot[Sin[x],{x, 0, 2 Pi}];
```

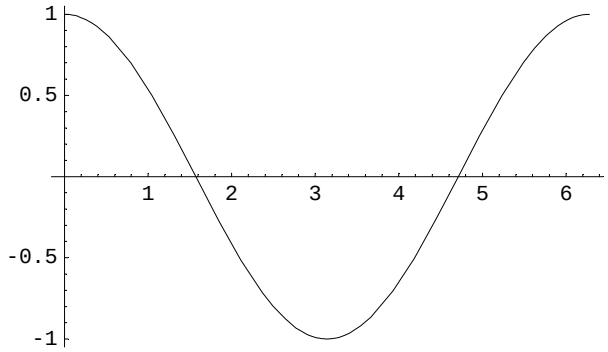
```
In[106] :> Plot[Sin[x],{x, 0, 2 Pi}];
```



In[107]:=

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

```
In[107] :> 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)

In[108]:=
```
(* Needs["Algebra`Trigonometry`"] old ! *)
```

In[109]:=
```
??TrigReduce
```

```
TrigReduce[expr] rewrites products and powers of trigonometric functions
   in expr in terms of trigonometric functions with combined arguments. Mehr…

Attributes[TrigReduce] = {Protected}

Options[TrigReduce] = {Modulus → 0}
```

In[110]:=
```
?Trig*
```

System`

```
Trig       TrigFactor     TrigReduce
TrigExpand TrigFactorList TrigToExp
```

In[111]:=
```
(* ??TrigReduceRel Old *)
```

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:

In[112]:=
```
TrigReduce[Sin[x + Pi]]
```

Out[112]=
```
-Sin[x]
```

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 chose sera expliquée dans un chapitre ultérieur. Ici il n'y a que du texte - 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 valeurs de règles d'assignation

Studiere die folgende Anwendung:

■ Etudie l'application suivante:

In[113]:=

```
Needs["Statistics`DescriptiveStatistics`"]
```

In[114]:=

```
dReport = DispersionReport[{3,4,2,3,6,7,4,8,9,
                            5,2,4}]
```

Out[114]=

$$\left\{\texttt{Variance} \to \frac{233}{44},\ \texttt{StandardDeviation} \to \frac{\sqrt{\frac{233}{11}}}{2},\ \texttt{SampleRange} \to 7,\right.$$
$$\left.\texttt{MeanDeviation} \to \frac{15}{8},\ \texttt{MedianDeviation} \to \frac{3}{2},\ \texttt{QuartileDeviation} \to \frac{7}{4}\right\}$$

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:

In[115]:=

```
MeanDeviation /. dReport
```

Out[115]=

$$\frac{15}{8}$$

Oder wir können das Resultat in eine Variable speichern:

■ Oubien nous pouvons mémoriser le résultat dans une variable:

In[116]:=
```
x1 = MeanDeviation /. dReport
```

Out[116]=
$$\frac{15}{8}$$

In[117]:=
```
x1
```

Out[117]=
$$\frac{15}{8}$$

7.3.8. Weiterrechnen mit Werten aus Output in Form von Ersetzungsregeln
▮ Continuer de calculer avec des valeurs de l'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:

In[118]:=
```
Remove[x]
```

In[119]:=
```
wurzeln = Solve[x^2 + 13x - 34 == 0]
```

Out[119]=
$$\left\{\left\{x \to \frac{1}{2}\left(-13-\sqrt{305}\right)\right\}, \left\{x \to \frac{1}{2}\left(-13+\sqrt{305}\right)\right\}\right\}$$

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:

In[120]:=
```
x^2 + 13x - 34 == 0 /. wurzeln
```

```
N::meprec : Internal precision limit $MaxExtraPrecision =
   49.99999999999999` reached while evaluating -34 + 13/2 (-13 - Sqrt[305]) + 1/4 (-13 - Sqrt[305])^2. Mehr…

N::meprec : Internal precision limit $MaxExtraPrecision =
   49.99999999999999` reached while evaluating -34 + 13/2 (-13 + Sqrt[305]) + 1/4 (-13 + Sqrt[305])^2. Mehr…
```

Out[120]=
$$\left\{-34+\frac{13}{2}\left(-13-\sqrt{305}\right)+\frac{1}{4}\left(-13-\sqrt{305}\right)^2 == 0,\ -34+\frac{13}{2}\left(-13+\sqrt{305}\right)+\frac{1}{4}\left(-13+\sqrt{305}\right)^2 == 0\right\}$$

In[121]:=
```
Simplify[%]
```

Out[121]=
```
{True, True}
```

Oder: ■ Ou:

In[122]:=
```
x /. wurzeln[[1]]
```

Out[122]=
$$\frac{1}{2}\left(-13-\sqrt{305}\right)$$

Die erste Regel wurde extrahiert! Oder:
■ Cette règle a été extraite! Ou:

In[123]:=
```
2x + 13 /. wurzeln[[1]]
```

Out[123]=
$$-\sqrt{305}$$

7.4. Gleichheit
■ Egalité

7.4.1. Allgemeines
■ Généralités

Studiere: ■ Etudie:

In[124]:=
```
?==
```

```
lhs == rhs returns True if lhs and rhs are identical. Mehr…
```

Anwendung: ■ Application:

In[125]:=
```
Expand[(x + y)^2] == x^2 + 2x y + y^2
```

Out[125]=
```
True
```

Vergleich mit "=":
■ Comparaison avec "=":

In[126]:=
```
Expand[(x + y)^2] =  x^2 + 2x y + y^2
```

```
Set::write : Tag Expand in Expand[(x + y)^2] is Protected. Mehr…
```

Out[126]=
$$x^2 + 2\,x\,y + y^2$$

Eingebaute Funktionen sind "protected". Man kann dies allerdings ändern. Beispiel:

■ Les fonctions incorporées sont "prodected". Mais on peut changer cela. Exemple:

In[127]:=
```
?Out
```
```
%n or Out[n] is a global object that is assigned to be the value produced
   on the nth output line. % gives the last result generated. %% gives the
   result before last. %% ... % (k times) gives the kth previous result. Mehr…
```

In[128]:=
```
Out = 1
```
```
Set::wrsym : Symbol Out is Protected. Mehr…
```

Out[128]=
```
1
```

In[129]:=
```
Attributes[Out]
```

Out[129]=
```
{Listable, Protected}
```

Achtung: Das gibt viel Output..... Vgl. 7.1.4.

■ Attention: Cela fait beaucoup d'output.... Compare 7.1.4.

In[130]:=
```
(*  ??Out   *)
```

In[131]:=
```
?Unprotect
```
```
Unprotect[s1, s2, ... ] removes the attribute Protected
   for the symbols si. Unprotect["form1", "form2", ... ] unprotects
   all symbols whose names textually match any of the formi. Mehr…
```

In[132]:=
```
Unprotect[Out]
```

Out[132]=
```
{Out}
```

In[133]:=
```
Out = 2
```

Out[133]=
```
2
```

In[134]:=
```
Clear[Out]; Protect[Out]
```

Out[134]=
```
{Out}
```

Wieviel Output ist es noch?

■ Combien d'output cela fait-il encore?

In[135]:=

```
Out = 3
```

```
Set::wrsym : Symbol Out is Protected. Mehr…
```

Out[135]=

```
3
```

7.4.2. Konvertierung von Gleichheiten in Zuweisungsregeln
▮ Convertir des églaités en règles d'assignation

Studiere: ▮ Etudie:

In[136]:=

```
?ToRules
```

```
ToRules[eqns] takes logical combinations of equations, in the form generated by Roots and
   Reduce, and converts them to lists of rules, of the form produced by Solve. Mehr…
```

In[137]:=

```
NRoots[x^3 - 5 x^2 - 19 x + 66 == 0 , x]
```

Out[137]=

```
x == -3.9708 || x == 2.61524 || x == 6.35557
```

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:

In[138]:=

```
ToRules[%]
```

Out[138]=

```
Sequence[{x → -3.9708}, {x → 2.61524}, {x → 6.35557}]
```

7.4.2. Weitere Probleme mit "Gleichheit"
▮ Autres problèmes avec "églaité"

Probiere aus: ▮ Essaie:

In[139]:=

```
?=
```

```
lhs = rhs evaluates rhs and assigns the result to be the value of lhs. From then on, lhs is
   replaced by rhs whenever it appears. {l1, l2, ... } = {r1, r2, ... } evaluates
   the ri, and assigns the results to be the values of the corresponding li. Mehr…
```

In[140]:=

```
?==
```

```
lhs == rhs returns True if lhs and rhs are identical. Mehr…
```

In[141]:=

```
?===
```

```
lhs === rhs yields True if the expression
   lhs is identical to rhs, and yields False otherwise. Mehr…
```

In[142]:=
```
?!=
```
lhs != rhs returns False if lhs and rhs are identical. Mehr...

In[143]:=
```
?=!=
```
lhs =!= rhs yields True if the expression
lhs is not identical to rhs, and yields False otherwise. Mehr...

In[144]:=
```
?Set
```
lhs = rhs evaluates rhs and assigns the result to be the value of lhs. From then on, lhs is replaced by rhs whenever it appears. {l1, l2, ... } = {r1, r2, ... } evaluates the ri, and assigns the results to be the values of the corresponding li. Mehr...

In[145]:=
```
?Equal
```
lhs == rhs returns True if lhs and rhs are identical. Mehr...

In[146]:=
```
?SameQ
```
lhs === rhs yields True if the expression
lhs is identical to rhs, and yields False otherwise. Mehr...

In[147]:=
```
?Unequal
```
lhs != rhs returns False if lhs and rhs are identical. Mehr...

In[148]:=
```
?UnsameQ
```
lhs =!= rhs yields True if the expression
lhs is not identical to rhs, and yields False otherwise. Mehr...

Studiere: ■ Etudie:

In[149]:=
```
N[Pi, 5] == N[Pi, 30]
```
Out[149]=
```
True
```

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.

In[150]:=
```
Clear[x]; Clear[y];
```
In[151]:=
```
x == y
```
Out[151]=
```
x == y
```

Hier gab es nichts zu prüfen.
■ Ici il n'y avait rien à examiner.

```
In[152]:=
        1 != 2
Out[152]=
         True
```

Das stimmt. ■ C'est vrai.

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.

```
In[153]:=
        N[Pi, 3] === N[Pi, 30]
Out[153]=
         True
```

Oder: ■ Ou:

```
In[154]:=
        N[Pi, 3] =!= N[Pi, 30]
Out[154]=
         False

In[155]:=
        x =!= y
Out[155]=
         True

In[156]:=
        x === y
Out[156]=
         False
```

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

```
In[157]:=
        (* Old Form: Remove["Global`@*"] *)

In[158]:=
         Remove["Global`*"]
```