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.

Run mit WIN+*Mathematica* Version 5.2
▮ Testé avec *Mathematica* version 5.2+WIN

13.1. Fehlermeldungen
▮ Annonces d'erreurs

Beispiel: ▮ Exemples:

```
In[1]:= 1 + 2 = 3

        Set::write : Tag Plus in 1 + 2 is Protected. Mehr…

Out[1]= 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.(Les 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 le peuvent pas.

Beispiel: ▮ Exemple:

In[2]:= **2 -)5 + 7**

```
Syntax::bktx: "2 - " has extra ")" after it. Mehr…
```

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 examen des parenthèses. Essaie (met le cursor dans l'espace vide et emploie (<Command>-B (B grand) --- NeXT) <Shift Control> B (B grand)):

In[2]:= **((([[cjklf]r]}}]**

```
Syntax::bktmch:
   "([ [cjklf]r]" must be followed by ")", not "}". Mehr…
```

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:

In[2]:=
```
t = 5;
Print["t = ", t];
Integrate[Sin[t], t]
```

```
t = 5

Integrate::ilim : Invalid integration variable or limit(s) in 5. Mehr…
```

Out[4]= $\int \text{Sin}[5]\, d5$

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:

In[5]:= **D[x^3]**

Out[5]= x^3

```
In[6]:= D[x^3, x]

Out[6]= 3 x^2

In[7]:= D[x^3, y]

Out[7]= 0

In[8]:= D[x^3, 1]

        General::ivar : 1 is not a valid variable. Mehr…

Out[8]= ∂1 x^3

In[9]:= D[x^3 Sin[y], x, y]

Out[9]= 3 x^2 Cos[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:

```
In[10]:= Clear[f];
         f[x_] := If[x > 0,  x -x];
         {f[5], f[-5]}

Out[12]= {0, Null}
```

Mit Komma:
Avec virgule:

```
In[13]:= Clear[f];
         f[x_] := If[x > 0,  x, -x];
         {f[5], f[-5]}

Out[15]= {5, 5}
```

Studiere: Etudie:

```
In[16]:= Clear[f];
         f[x_, y_] := xy;
         f[2, 3]

Out[18]= xy

In[19]:= Clear[f];
         f[x_, y_] := x y;
         f[2, 3]

Out[21]= 6

In[22]:= D[x^3 Sin[y], x, y]

Out[22]= 3 x^2 Cos[y]
```

13.4. Falsche Klammerung
▮ Erreurs de parenthèses

Beispiel: ▮ Exemple:

```
In[23]:= Clear[g]

In[24]:= g[0]

Out[24]= g[0]

In[25]:= g(0)

Out[25]= 0

In[26]:= g(a)

Out[26]= a g
```

13.5. Verwechslung von Zeilen- und Spaltenvektor
▮ Confusion entre vecteur de ligne et vecteur de colonnes

Skalarprodukt
▮ Produit scalaire

```
In[27]:= {a, b, c}.{e, f, g}

Out[27]= a e + b f + c g

In[28]:= {a, b, c}.{e, f, g, h}

        Dot::dotsh : Tensors {a, b, c} and {e, f, g, h} have incompatible shapes. Mehr…

Out[28]= {a, b, c}.{e, f, g, h}
```

Matrixprodukt
▮ Produit de matrice

Uebersicht: ▮ Vue d'ensemble:

```
In[29]:= {{a},{b},{c}} // MatrixForm

Out[29]//MatrixForm=
        ⎛a⎞
        ⎜b⎟
        ⎝c⎠

In[30]:= Transpose[{{a},{b},{c}}] // MatrixForm

Out[30]//MatrixForm=
        (a  b  c)
```

Diverse Produkte:
■ Produits divers:

```
In[31]:= {{a},{b},{c}} . {{e},{f},{g}} // MatrixForm

        Dot::dotsh : Tensors {{a}, {b}, {c}} and {{e}, {f}, {g}} have incompatible shapes. Mehr…

Out[31]//MatrixForm=
        {{a}, {b}, {c}}.{{e}, {f}, {g}}
```

```
In[32]:= {{a},{b},{c}} . Transpose[{{e},{f},{g}}] //
                                MatrixForm
```

Out[32]//MatrixForm=

$$\begin{pmatrix} a\,e & a\,f & a\,g \\ b\,e & b\,f & b\,g \\ c\,e & c\,f & c\,g \end{pmatrix}$$

```
In[33]:= Transpose[{{a},{b},{c}}] . {{e},{f},{g}} //
                                MatrixForm
```

Out[33]//MatrixForm=

$$(a\,e + b\,f + c\,g)$$

```
In[34]:= Transpose[{{a},{b},{c}}] .
             Transpose[{{e},{f},{g}}] // MatrixForm

        Dot::dotsh : Tensors {{a, b, c}} and {{e, f, g}} have incompatible shapes. Mehr…

Out[34]//MatrixForm=
        {{a, b, c}}.{{e, f, g}}
```

Generierung: ■ Génération:

```
In[35]:= Table[u[[i]] v[[j]], {i, Length[u]}, {j,
                                Length[v]}]

Out[35]= {}
```

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 "Enter"

Beispiele ■ Exemple

```
In[36]:= a = 5
           + 7
           + 9

Out[36]= 5

Out[37]= 7

Out[38]= 9
```

```
In[39]:= a

Out[39]= 5

In[40]:= a = 5 +
           7 +
           9

Out[40]= 21

In[41]:= a

Out[41]= 21
```

Was ist hier los?

■ Que se passe-t-il ici?

13.7. Probleme mit Wurzeln (Mehrdeutigkeiten)
■ Problèmes de racines (différentes acceptions, équivoques)

Beispiele
■ Exemples

```
In[42]:= (x^2)^(0.5)

Out[42]= (x^2)^0.5

In[43]:= Sqrt[x^2]

Out[43]= Sqrt[x^2]

In[44]:= Integrate[Sqrt[1 - Cos[x]^2], {x, 0, Pi/2}]

Out[44]= 1

In[45]:= Integrate[Cos[x], {x, Pi, 2Pi}]

Out[45]= 0

In[46]:= Integrate[Sqrt[1 - Sin[x]^2], {x, Pi, 2Pi}]

Out[46]= 2
```

Was ist hier los?

■ Que se passe-t-il ici?

```
In[47]:= ??Sqrt

         Sqrt[z] gives the square root of z. Mehr...

         Attributes[Sqrt] = {Listable, NumericFunction, Protected}

In[48]:= Sqrt[(-2)^2]

Out[48]= 2
```

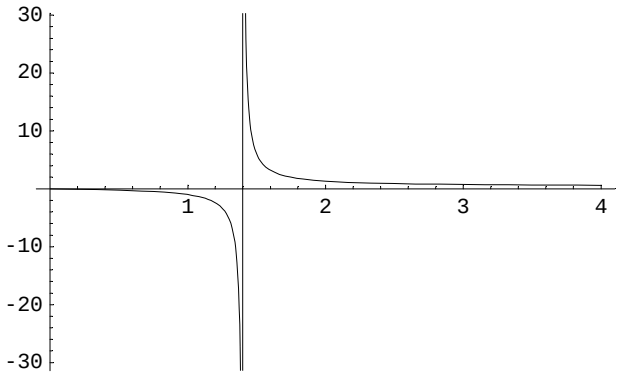
13.8. Abarbeitungsreihenfolge
▮ Ordre de travail

Probiere: ▮ Essaie:

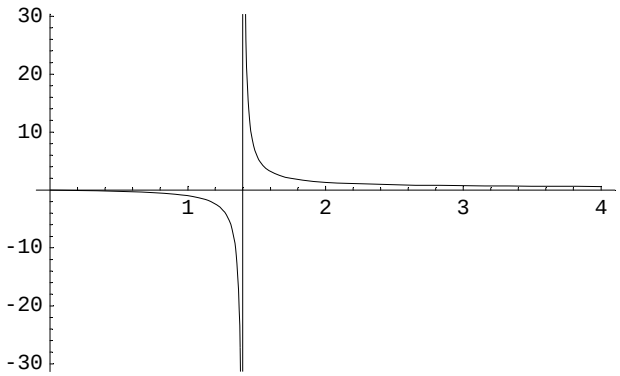
```
In[49]:= s = 2 x /(5x - 7)
```

$$\text{Out[49]= } \frac{2\,x}{-7+5\,x}$$

```
In[50]:= Plot[s, {x, 0, 4}];
```

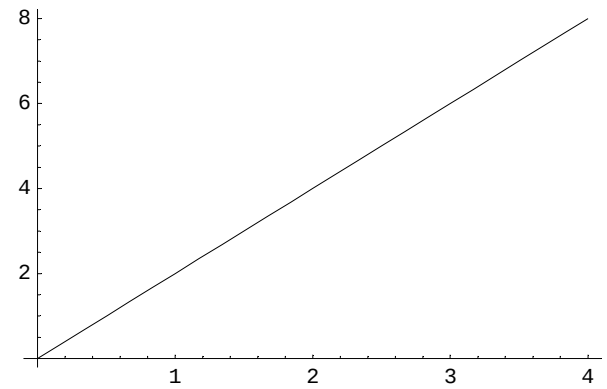



```
In[51]:= Plot[Numerator[s], {x, 0, 4}];
```



Wieso der Fehler? Probiere:
▮ Pourquoi l'erreur? Essaie:

```
In[52]:= 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!

```
In[53]:=
        Clear[f];
        f[x_] := x^2;
        f[y_] := y^3;

In[56]:= {f[x],f[y]}

Out[56]= {x^3, y^3}

In[57]:= ??f

       Global`f

        f[y_] := y^3
```

Beispiel: ▮ Exemple:

```
In[58]:= Clear[ung];
         ung[x_?OddQ ] := (ung[x] = 1);
         ung[x_?EvenQ] := (ung[x] = 0);


In[61]:= Table[ung[x], {x,10}]

Out[61]= {1, 0, 1, 0, 1, 0, 1, 0, 1, 0}

In[62]:= ??ung

       Global`ung

        ung[1] = 1

        ung[2] = 0

        ung[3] = 1

        ung[4] = 0

        ung[5] = 1

        ung[6] = 0

        ung[7] = 1

        ung[8] = 0

        ung[9] = 1

        ung[10] = 0

        ung[x_?OddQ] := ung[x] = 1

        ung[x_?EvenQ] := ung[x] = 0
```

13.10. "Protect" und Funktionen
■ "Protect" et fonctions

In[63]:= **??Protect**

```
Protect[s1, s2, ... ] sets the attribute Protected
   for the symbols si. Protect["form1", "form2", ... ] protects all
   symbols whose names match any of the string patterns formi. Mehr...

Attributes[Protect] = {HoldAll, Protected}
```

In[64]:= **??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...

Attributes[Unprotect] = {HoldAll, Protected}
```

Probiere: ■ Essaie:

In[65]:= **Integrate[f[x_], x_] := F[x]**

SetDelayed::write : Tag Integrate in $\int x\_^3 \partial x\_$ is Protected. Mehr…

Out[65]= $Failed

In[66]:=
```
Remove[f, F];
f/: Integrate[f[x_], x_] := F[x];
Integrate[f[x], x]
```

Out[68]= F[x]

Probiere: ■ Essaie:

In[69]:= **Pi = 3.1416**

Set::wrsym : Symbol π is Protected. Mehr…

Out[69]= 3.1416

In[70]:= **{Pi, Pi // N}**

Out[70]= {π, 3.14159}

In[71]:=
```
Unprotect[Pi];
Pi = 3.1416;
Pi
```

Out[73]= 3.1416

In[74]:=
```
Protect[Pi];
Pi // N
```

Out[75]= 3.1416

In[76]:= **??Pi**

Pi is the constant pi, with numerical value approximately equal to 3.14159. Mehr...

Attributes[π] = {Constant, Protected, ReadProtected}

In[77]:=
```
Unprotect[Pi];
Remove[Pi];
Pi // N
```

Out[79]= Pi

In[80]:= **??Pi**

Pi is the constant pi, with numerical value approximately equal to 3.14159.

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'une 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 de problèmes en travaillant seul.

Studiere: ∎ Etudie:

In[81]:= **?On**

```
On[symbol::tag] switches on a message, so that it can be printed.
   On[s] switches on tracing for the symbol s. On[m1, m2, ... ] switches
   on several messages. On[ ] switches on tracing for all symbols. Mehr...
```

In[82]:= **?Off**

```
Off[symbol::tag] switches off a message, so that it is no longer printed. Off[s]
   switches off tracing messages associated with the symbol s. Off[m1, m2, ... ]
   switches off several messages. Off[ ] switches off all tracing messages. Mehr...
```

In[83]:= **?Print**

```
Print[expr1, expr2, ... ] prints the expri concatenated together. Mehr...
```

In[84]:= **?Debug**

```
Debug is an obsolete function, superseded by Trace. Debug[expr] evaluates
  expr, allowing you to stop at certain points in some control structures,
  and see what's happening or run commands. Debug[expr, {f1, f2, ...}]
  stops only in evaluation of the functions fi (and everything they call).
```

In[85]:= **?Trace**

```
Trace[expr] generates a list of all expressions used in the evaluation of expr. Trace[expr,
   form] includes only those expressions which match form. Trace[expr, s] includes
   all evaluations which use transformation rules associated with the symbol s. Mehr...
```

In[86]:= **?Input**

```
Input[ ] interactively reads in one Mathematica expression.
   Input["prompt"] requests input, using the specified string as a prompt. Mehr...
```

In[87]:= **?Interrupt**

```
Interrupt[ ] generates an interrupt. Mehr...
```

13.11.2. Traceing

▮ Traceing

Probiere aus: ▮ Essaie:

In[88]:=
```
Remove[f, g];
f[x_]:= x^3 + g[x];
g[x_]:= x - 14
```

In[91]:= **On[f, g]**

In[92]:= **f[4]**

```
f::trace : f[4] --> 4^3 + g[4]. Mehr…

g::trace : g[4] --> 4 - 14. Mehr…
```

Out[92]= 54

In[93]:= **Off[f, g]**

In[94]:= **f[4]**

Out[94]= 54

13.11.3. Print
▮ Print

Von 10.3.7:
▮ De 10.3.7:

```
In[95]:= 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}]
```

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

Länge der Liste: 6

Sortierte Liste: {24, 32, 36, 47, 49, 52}

Median:
```

$$\text{Out[96]=}\ \frac{83}{2}$$

13.11.4. Debug (für Version 1.2)
▮ Debug (pour version 1.2)

Beispiel: ▮ Exemple:

```
In[97]:= ?$Version

$Version is a string that represents the version of Mathematica you are running. Mehr…

In[98]:= ?$VersionNumber

$VersionNumber is a real number which gives the current Mathematica
   kernel version number, and increases in successive versions. Mehr…

In[99]:= $VersionNumber < 2

Out[99]= False
```

In[100]:=
```
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!"]]
```

```
Achtung: Bei Debug ab Version 2:        Systemabsturz!
  Attention: Debug dès version 2:               système interrompu!
```

13.11.5. Trace

▮ Trace

Studiere: ▮ Etudie:

In[101]:=
```
2 5 + 8 9
```

Out[101]=
```
82
```

In[102]:=
```
Trace[2 5 + 8 9]
```

Out[102]=
```
{{2 5, 10}, {8 9, 72}, 10 + 72, 82}
```

In[103]:=
```
Trace[2 5 + 8 9, TraceDepth -> 1]
```

Out[103]=
```
{10 + 72, 82}
```

In[104]:=
```
Trace[2 5 + 8 9, Plus]
```

Out[104]=
```
{10 + 72, 82}
```

In[105]:=
```
Trace[2 5 + 8 9, Times]
```

Out[105]=
```
{{2 5, 10}, {8 9, 72}}
```

In[106]:=
```
Trace[2 5 + 8 9, x_ y_]
```

Out[106]=
```
{{2 5}, {8 9}}
```

In[107]:=
```
Trace[2 5 + 8 9, TraceDepth -> 1]
```

Out[107]=
```
{10 + 72, 82}
```

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 une fenêtre, tu peux demander le contenu d'une variable. Ecris p.ex. "?len" dans la fenêtre et observe ce qui se passe!

In[108]:=
```
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}]
```

```
Global`len

len = 6

Null
```

Out[109]=

$$\frac{83}{2}$$

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.

In[110]:=
```
Remove[f];
f::gross = "Eingegebene Zahl grösser tausend. \
        Rechenzeit möglicherweise sehr lang.";
f[n_] := If[n <= 500, Print[n,"! = ", n!],
        f::gross]
```

In[113]:=

```
f[10]
```

```
10! = 3628800
```

In[114]:=

```
f[100]
```

```
100! =
 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761
  565182862536979208272237582511852109168640000000000000000000000000
```

In[115]:=

```
f[500]
```

```
500! =
 1220136825991110068701238785423046926253574342803192842192413588385845373153881997605496447 5
  022032818630136164771482035841633787220781772004807852051593292854779075719393306037729608 5
  908627042917454788242491272634430567017327076946106280231045264421887878946575477714986349 4
  367781037644274033827365397471386477878495438489595537537990423241061271326984327745715546 3
  099772027810145610811883737095310163563244329870295638966289116589747695720879269288712817 8
  007026517450776841071962439039432253642260523494585012991857150124870696156814162535905669 3
  423813008856249246891564126775654481886506593847951775360894005745238940335798476363944905 3
  130623237490664450488246650759467358620746379251842004593696929810222639719525971909452178 2
  333175693458150855233282076282002340262690789834245171200620771464097945611612762914595123 7
  229913340169552363850942885592018727433795173014586357570828355780158735432768888680120399 8
  823847021514676054454076635359841744304801289383138968816394874696588175045069263653381750 5
  547812864000000000000000000000000000000000000000000000000000000000000000000000000000000000 0
  000000000000000000000000000000000000000000
```

In[116]:=

```
f[501]
```

Out[116]=

```
Eingegebene Zahl grösser tausend.   Rechenzeit möglicherweise sehr lang.
```

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

In[117]:=

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

In[118]:=

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