

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.

Run mit WIN+*Mathematica* Version 5.2

■ Testé avec *Mathematica* version 5.2+WIN

WIR94/98/99/2000/2007 // Copyright Rolf Wirz

Aufgabe 1 ■ Problème 1

Untersuche, was die unten definierte Funktion *f* macht:
■ Examine ce que fait la fonction *f* définie ci-dessous:

Definition: ■ Définition:

```
In[1]:= f[2] = 3 ;  
        f[u] := 2 u^2;  
        f[v_] := 1/v ;  
        u = 2;
```

Was ist *f*[*u*] ?

■ Qu'est-ce *f*[*u*] ?

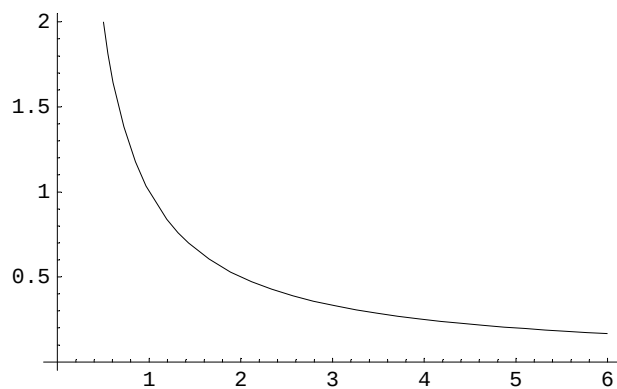
```
In[5]:= f[u]
```

```
out[5]= 3
```

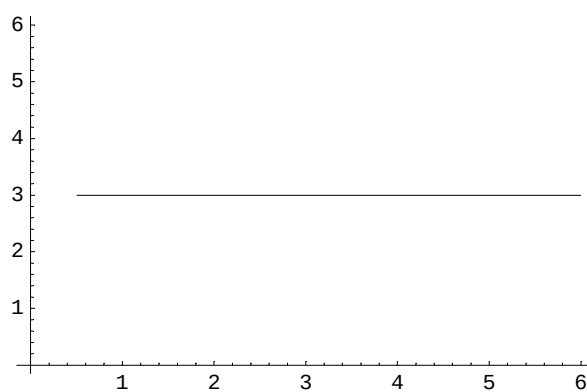
Was ist mit *f* passiert? Untersuche die möglichen Graphen!

Que s'est-il passé? Examine les graphes possibles!

```
In[6]:= Plot[f[x], {x, 0.5, 6}];
```



```
In[7]:= Plot[f[u], {x, 0.5, 6}];
```



```
In[8]:= f[2]
```

```
Out[8]= 3
```

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:

```
In[9]:= Clear[f];
        f[x_?OddQ] := (x+1)/2
```

Ausprobieren: ■ Essayer:

```
In[11]:= f[3.2]
```

```
Out[11]= f[3.2]
```

```
In[12]:= f[3]
```

```
Out[12]= 2
```

```
In[13]:= f[2]
```

```
Out[13]= 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 $\{\{1, 1\}, \{2, 2\}\}$.

```
In[14]:= Clear[verdoppelnY];  
verdoppelnY[{x_, y_}] := {x, 2y}
```

```
In[16]:= Map[verdoppelnY, {{1, 1}, {2, 2}}]
```

```
Out[16]= {{1, 2}, {2, 4}}
```

Was macht "Map" hier?

■ Que fait "Map" ici?

Aufgabe 4 ■ Problème 4

Definiere eine Funktion "dreieck[x]" wie folgt:

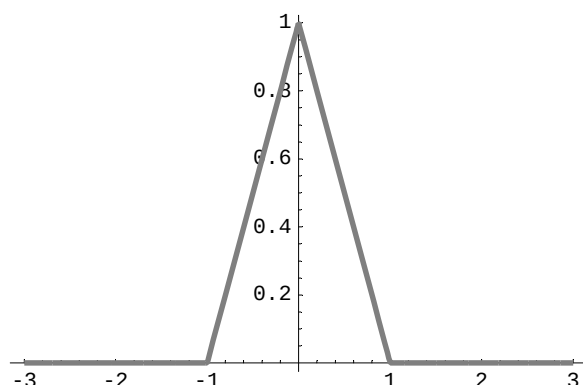
■ Définis une fonction "dreieck[x]"

dreieck[x] ist gleich $1 - \text{Abs}[x]$ für $x < 1$. Sonst ist der Funktionswert null.

Mache einen Plot!

■ dreieck[x] est égal $1 - \text{Abs}[x]$ pour $x < 1$. Autrement la valeur de la fonction est zéro. Fais un plot!

```
In[17]:= 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

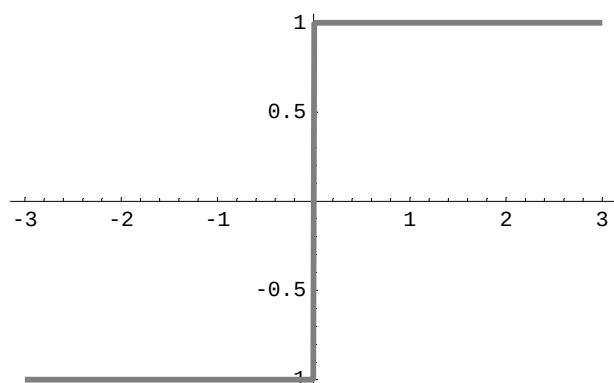
Definiere eine Funktion "signum[x]" wie folgt:

■ Définis une fonction "signum[x]" comme il suit:

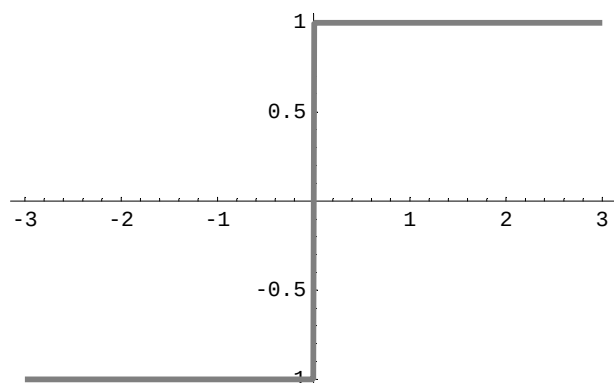
signum[x] ist gleich 1 für $x > 0$, -1 für $x < 0$ und 0 für $x = 0$. Mache einen Plot! (Definiere dazu signum für Floating-Point-Zahlen 0.0!) Schreibe einen Kommentar für die Dokumentation!

■ 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 virgule flottante 0.0!) Ecris un commentaire pour la documentation!

```
In[21]:= 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]}}];
```



```
In[27]:= 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]}}];
```



```
In[33]:= ?signum
```

```
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.
```

```
In[34]:= ??signum
```

```
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.
```

```
signum[0.] = 0.
```

```
signum[x_ /; x > 0] := 1
```

```
signum[x_ /; x < 0] := -1
```

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 $n+kd < m$.

```

In[35]:= 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]}];

{6, {1, 2, 3, 4, 5, 6}}

{-6, {}}

{0, {}}

{{5, 11}, {{1, 2, 3, 4, 5}, {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}}}

{5, 11, {5, 6, 7, 8, 9, 10, 11}}

{5, -3, {}}

{5, 3, {}}

{5, 18, 4, {5, 9, 13, 17}}

```

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?

```
In[47]:= sRege1 = s[-x_] :> -s[x]
```

```
Out[47]= s[-x_] :> -s[x]
```

```
In[48]:= s[-a] /. sRege1
```

```
Out[48]= -s[a]
```

```
In[49]:= s[-3] /. sRege1
```

```
Out[49]= s[-3]
```

Wieso geschieht das? Beobachte, wie *Mathematica* -a und -3 interpretiert:

■ Pourquoi cela arrive-t-il? Observe, comme *Mathematica* interprète -a et -3:

```
In[50]:= FullForm[-a]
```

```
Out[50]//FullForm=
Times[-1, a]
```

```
In[51]:= FullForm[-3]
```

```
Out[51]//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.

```
In[52]:= FullForm[1-x]
```

```
Out[52]//FullForm=
Plus[1, Times[-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_):

```
In[53]:= s[1-x] /. sRege1
```

```
Out[53]= s[1-x]
```

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:

$$2 \cdot 4 \cdot 4 \cdot 6 \cdot 6 \cdot 8 \cdot 8 \cdot 10 \cdot 10 \dots 2n / (3 \cdot 3 \cdot 5 \cdot 5 \cdot 7 \cdot 7 \cdot 9 \cdot 9 \dots (2n+1)).$$

Entwickle: ■ Développe:

$$\text{piDurchVier}[1] = 2/3$$

$$\text{piDurchVier}[2] = 2 \cdot 4 \cdot 4 / (3 \cdot 3 \cdot 5) = \text{piDurchVier}[1] \cdot 4 \cdot 4 / (3 \cdot 5)$$

$$\text{piDurchVier}[3] = 2 \cdot 4 \cdot 4 \cdot 6 \cdot 6 / (3 \cdot 3 \cdot 5 \cdot 5 \cdot 7)$$

$$= \text{piDurchVier}[2] \cdot 6 \cdot 6 / (5 \cdot 7)$$

$$\text{piDurchVier}[4] = \text{piDurchVier}[3] \cdot 8 \cdot 8 / (7 \cdot 9)$$

```

In[54]:= RemoveAll[piDurchVier];
          piDurchVier[1] = 2/3;
          piDurchVier[n_] :=
            piDurchVier[n-1] (2n)^2 / ((2n-1)(2n+1));
          ??piDurchVier

Global`piDurchVier

piDurchVier[1] =  $\frac{2}{3}$ 

piDurchVier[n_] :=  $\frac{\text{piDurchVier}[n-1] (2n)^2}{(2n-1) (2n+1)}$ 

In[58]:= Table[piDurchVier[i],{i,5}]

Out[58]= {  $\frac{2}{3}$ ,  $\frac{32}{45}$ ,  $\frac{128}{175}$ ,  $\frac{8192}{11025}$ ,  $\frac{32768}{43659}$  }

In[59]:= Table[piDurchVier[i],{i,5}] // N

Out[59]= {0.666667, 0.711111, 0.731429, 0.743039, 0.750544}

Kontrolle: ■ Contrôle:

In[60]:= N[Pi/4,10]

Out[60]= 0.7853981634

In[61]:= N[piDurchVier[200]]

Out[61]= 0.784419

```

Was hältst Du von der Sache?

■ Qu'en penses-tu?

Aufgabe 9 ■ Problème 9

Spezielle Funktionen bestimmen:

■ Déterminer des fonctions spéciales:

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"?

```

In[62]:= ??Names

Names["string"] gives a list of the names of symbols which match the string. Names["string",
  SpellingCorrection->True] includes names which match after spelling correction. Mehr...

Attributes[Names] = {Protected}

Options[Names] = {IgnoreCase->False, SpellingCorrection->False}

```


In[63]:= ??MemberQ

MemberQ[list, form] returns True if an element of list matches form, and False otherwise.

MemberQ[list, form, levelspec] tests all parts of list specified by levelspec. Mehr...

Attributes[MemberQ] = {Protected}

Options[MemberQ] = {Heads → False}

In[64]:= ??Attributes

Attributes[symbol] gives the list of attributes for a symbol. Mehr...

Attributes[Attributes] = {HoldAll, Listable, Protected}

In[65]:= ??Select

Select[list, crit] picks out all elements ei of list for which crit[ei] is True. Select[list, crit, n] picks out the first n elements for which crit[ei] is True. Mehr...

Attributes[Select] = {Protected}

Suche alle "Listablen" Funktionen:

■ Cherche toutes les fonctions "Listables":

```
In[66]:= RemoveAll[listbareQ];
listbareQ[symbol_String]:=
MemberQ[Attributes[symbol], Listable]
```

Was tun die Teile?

■ Qu'en font les parties?

In[68]:= Attributes["Cos"]

Out[68]= {Listable, NumericFunction, Protected}

In[69]:= MemberQ[Attributes["Cos"], Listable]

Out[69]= True

In[70]:= listbareQ["Cos"]

Out[70]= True

Versuchen wir, die Funktionen zu bestimmen:

■ Essayons de déterminer les fonctions:

In[71]:= Select[Names["A* B* C* D* E*"], listbareQ]

Out[71]= {}

In[72]:= Select[Names["A*"], listbareQ]

Out[72]= {Abs, AiryAi, AiryAiPrime, AiryBi, AiryBiPrime, Apart, ArcCos, ArcCosh, ArcCot, ArcCoth, ArcCsc, ArcCsch, ArcSec, ArcSech, ArcSin, ArcSinh, ArcTan, ArcTanh, Arg, ArithmeticGeometricMean, Attributes}

In[73]:= Select[Names["B*"], listbareQ]

Out[73]= {BernoulliB, BesselI, BesselJ, BesselK, BesselY, Beta, BetaRegularized, Binomial, BitAnd, BitNot, BitOr, BitXor, Boole}

```

In[74]:= Select[Names["F*S*"], listbareQ]

Out[74]= {FactorSquareFree}

In[75]:= sN[x_]:=Select[Names[x], listbareQ]

In[76]:= sN["A*"]

Out[76]= {Abs, AiryAi, AiryAiPrime, AiryBi, AiryBiPrime, Apart, ArcCos,
  ArcCosh, ArcCot, ArcCoth, ArcCsc, ArcCsch, ArcSec, ArcSech, ArcSin,
  ArcSinh, ArcTan, ArcTanh, Arg, ArithmeticGeometricMean, Attributes}

In[77]:= Union[sN["A*"], sN["B*"], sN["C*"], sN["D*"],
  sN["E*"]]

Out[77]= {Abs, AiryAi, AiryAiPrime, AiryBi, AiryBiPrime, Apart, ArcCos, ArcCosh, ArcCot,
  ArcCoth, ArcCsc, ArcCsch, ArcSec, ArcSech, ArcSin, ArcSinh, ArcTan, ArcTanh,
  Arg, ArithmeticGeometricMean, Attributes, BernoulliB, BesselI, BesselJ,
  BesselK, BesselY, Beta, BetaRegularized, Binomial, BitAnd, BitNot, BitOr,
  BitXor, Boole, Cancel, CarmichaelLambda, Ceiling, Characters, ChebyshevT,
  ChebyshevU, Coefficient, Conjugate, ContinuedFraction, Cos, Cosh, Cot, Coth,
  CreateDirectory, Csc, Csch, Decompose, Denominator, DiracDelta, Divide,
  Divisors, DivisorSigma, EllipticE, EllipticF, EllipticK, EllipticPi,
  EllipticTheta, EllipticThetaPrime, Erf, Erfc, Erfi, EulerE, EulerPhi,
  EvenQ, Exp, ExpIntegralE, ExpIntegralEi, Exponent, ExpToTrig, ExtendedGCD}

```

"Putzmaschine" einsetzen

■ Employer la "machine de nettoyage"

```

In[78]:= (* Old Form: Remove["Global`@*"] *)

In[79]:= Remove["Global`*"]

```