


```
lil = Map[gamma, li]
General::spell1 :
Possible spelling error: new
symbol name "gamma" is similar
to existing symbol "Gamma".
| {gamma[1], gamma[π],
  gamma[5], gamma[7], gamma[3]}
lil /. gamma[n_] → Factorial[n - 1]
| {1, (-1 + π)!, 24, 720, 2}
lil /.
gamma[n_Integer] → Factorial[n - 1]
| {1, gamma[π], 24, 720, 2}
LineareFunktion[u_Plus] :=
Map[LineareFunktion, u]
LineareFunktion[5 a + 7 b + 8 c]
| LineareFunktion[5 a] +
  LineareFunktion[7 b] +
  LineareFunktion[8 c]
```

Bedingte Patterns

```
MyBinomial[n_Integer /; n > 0,
k_Integer /; k > 0] :=
n / k MyBinomial[n - 1, k - 1]
MyBinomial[5, 3]
| 10  $\binom{2}{0}$ 
MyBinomial[n_Integer /; n ≥ 0, 0] = 1
| 1
MyBinomial[5, 3]
| 10
MyBinomial[3, 5]
|  $\frac{1}{10} \binom{0}{2}$ 
MyBinomial[0, k_Integer /; k > 0] = 0
| 0
MyBinomial[5, 3]
| 10
MyBinomial[3, 5]
| 0
MyBinomial[n, k]
|  $\binom{n}{k}$ 
```

```
?? MyBinomial
Global`MyBinomial
(n_Integer /; n > 0) :=  $\frac{n \binom{-1+n}{-1+k}}{k}$ 
(n_Integer /; n ≥ 0) = 1
 $\binom{0}{k\_Integer /; k > 0} = 0$ 
MakeBoxes[ $\binom{n}{k}$ , FormatType_] ^:=
Format[MatrixForm[{{n}, {k}}],
FormatType]
Format[ $\binom{n}{k}$ ] := MatrixForm[{{n}, {k}}
```

Ein paar nützliche "Test"-Funktionen

```
liste = {-5, 1, 2, Pi, E, 2/3, 4.5, I}
| {-5, 1, 2, π, e,  $\frac{2}{3}$ , 4.5, i}
TesteFunktion[u_] :=
Map[{{#, u[#], Head[#]} &, liste] //
TableForm
TesteFunktion[NumberQ]
| -5 True Integer
  1 True Integer
  2 True Integer
  π False Symbol
  e False Symbol
   $\frac{2}{3}$  True Rational
  4.5 True Real
  i True Complex
TesteFunktion[IntegerQ]
| -5 True Integer
  1 True Integer
  2 True Integer
  π False Symbol
  e False Symbol
   $\frac{2}{3}$  False Rational
  4.5 False Real
  i False Complex
```

```
TesteFunktion[EvenQ]
| -5 False Integer
  1 False Integer
  2 True Integer
  π False Symbol
  e False Symbol
   $\frac{2}{3}$  False Rational
  4.5 False Real
  i False Complex
TesteFunktion[OddQ]
| -5 True Integer
  1 True Integer
  2 False Integer
  π False Symbol
  e False Symbol
   $\frac{2}{3}$  False Rational
  4.5 False Real
  i False Complex
TesteFunktion[PrimeQ]
| -5 True Integer
  1 False Integer
  2 True Integer
  π False Symbol
  e False Symbol
   $\frac{2}{3}$  False Rational
  4.5 False Real
  i False Complex
TesteFunktion[Element[#, Reals] &]
| -5 True Integer
  1 True Integer
  2 True Integer
  π True Symbol
  e True Symbol
   $\frac{2}{3}$  True Rational
  4.5 True Real
  i False Complex
```

```
TesteFunktion[
Element[#, Complexes] &]
| -5 True Integer
  1 True Integer
  2 True Integer
  π True Symbol
  e True Symbol
   $\frac{2}{3}$  True Rational
  4.5 True Real
  i True Complex
LineareFunktion[
n_x_ /; Element[n, Reals]] :=
n LineareFunktion[x]
LineareFunktion[a + Pi b + 7 c + d e]
| LineareFunktion[a] +
  π LineareFunktion[b] +
  7 LineareFunktion[c] +
  LineareFunktion[d e]
```

Variable Anzahl von Argumenten

```
Mittelwert[u_] :=
Apply[Plus, {u}] / Length[{u}]
Mittelwert[a, b, c]
|  $\frac{1}{3} (a + b + c)$ 
```

Programmierung

Speichern von Zwischenergebnissen

```
Fibo[n_Integer /; n > 1] =
Fibo[n - 1] + Fibo[n - 2]
Fibo[0] = 0
Fibo[1] = 1
| Fibo[-2 + n] + Fibo[-1 + n]
| 0
| 1
Map[Fibo, Range[10]]
| {1, 1, 2, 3, 5, 8, 13, 21, 34, 55}
Fibo[25]
| 75025
Clear[Fibo]
Fibo[n_Integer /; n > 1] :=
Fibo[n] = Fibo[n - 1] + Fibo[n - 2]
Fibo[0] = 0
Fibo[1] = 1
| 0
| 1
Fibo[25]
| 75025
```

Lokale Variablen

```
t = 17
| 17
Module[{t},
  t = 8;
  Print[t]]
8

t
| 17
Standardabweichung[x_List] :=
Module[{n, μ, σ},
  n = Length[x];
  μ = Plus @@ x/n;
  σ =
  Sqrt[Plus@@Map[(# - μ)^2 &, x] /
    (n - 1)];
  {μ, σ}
]
Standardabweichung[{a, b}]
| {
  {

$$\frac{a+b}{2}, \sqrt{\left(\left(a + \frac{1}{2}(-a-b)\right)^2 + \left(\frac{1}{2}(-a-b) + b\right)^2\right)}$$

}
}
Table[Random[], {100}]
Standardabweichung[%]
| {0.563602, 0.288844}
```

Do, For, While, If,

■ Do

```
Do[Print["Hallo"], {5}]
Hallo
Hallo
Hallo
Hallo
Hallo
Do[Print[i], {i, 5}]
1
2
3
4
5
```

```
Do[Print[i], {i, 1, 5, 1/2}]
1
| 3/2
2
| 5/2
3
| 7/2
4
| 9/2
5
```

■ For

```
For[i = 1, i ≤ 5, i = i + 1/2,
  Print[i]]
1
| 3/2
2
| 5/2
3
| 7/2
4
| 9/2
5
```

■ If

```
Do[If[Random[] < 0.5,
  Print["Kleiner 0.5"],
  Print["Nicht kleiner 0.5"]], {5}]
Kleiner 0.5
Nicht kleiner 0.5
Nicht kleiner 0.5
Kleiner 0.5
Nicht kleiner 0.5
```

```
If[Random[] < 0.5,
  (Print["Kleiner"];
  Print["als"]);
  Print["0.5"]],
(Print["Größer"];
  Print["als"];
  Print["0.5"])]
Größer
als
0.5
```

■ While

```
xalt = 2; xneu = 1
| 1
While[Abs[xalt - xneu] > 10^(-20),
  (xalt = xneu;
  xneu = 1/2 (xalt + 2/xalt))]
xneu
| 1572584048032918633353217
| 1111984844349868137938112
N[%]
| 1.41421
xneu - Sqrt[2] // N
| -2.22045 × 10-16
```

"Benutzerdefinierte Datentypen"

```
Format[Intervall[a_, b_]] :=
StringJoin[" ", ToString[a],
  ", ", ToString[b], ""]
General::spell1 :
Possible spelling error: new symbol
name "Intervall" is similar
to existing symbol "Interval".

Intervall[1, 5]
| [1, 5]
Intervall[1, 5] + Intervall[2, 6]
| [1, 5] + [2, 6]
Intervall[a_, b_] +
  Intervall[c_, d_] :=
  Intervall[a + c, b + d]
SetDelayed::write : Tag Plus in
[a_, b_] + [c_, d_] is Protected.

| $Failed
?? Intervall
Global`Intervall

MakeBoxes[[a_, b_], FormatType_] ^:=
Format[[<> ToString[a] <>, <>
  ToString[b] <>], FormatType]

Format[[a_, b_]] :=
[<> ToString[a] <>, <> ToString[b] <>]
```

```
Intervall[a_, b_] +
  Intervall[c_, d_] ^:=
  Intervall[a + c, b + d]
?? Intervall
Global`Intervall

[a_, b_] + [c_, d_] ^:= [a + c, b + d]

MakeBoxes[[a_, b_], FormatType_] ^:=
Format[[<> ToString[a] <>, <>
  ToString[b] <>], FormatType]

Format[[a_, b_]] :=
[<> ToString[a] <>, <> ToString[b] <>]

Intervall[1, 5] + Intervall[2, 6]
| [3, 11]
Intervall[a_, b_]
  Intervall[c_, d_] ^:=
  Intervall[a c, b d] /; c ≥ 0 && a ≥ 0
Intervall[1, 5] * Intervall[2, 6]
| [2, 30]
Intervall[a_, b_]
  Intervall[c_, d_] ^:=
  Intervall[c b, a d] /; a ≥ 0 && d ≤ 0
Intervall[1, 5] * Intervall[-6, -2]
| [-30, -2]
Intervall[-6, -2] * Intervall[1, 5]
| [-30, -2]
?? Times

x*y*z or x y z represents
a product of terms. More...

Attributes[Times] = {Flat, Listable, Nume:

Default[Times] := 1

?? Orderless

Orderless is an attribute that can
be assigned to a symbol f to
indicate that the elements ei in
expressions of the form f[e1,
e2, ...] should automatically
be sorted into canonical order.
This property is accounted
for in pattern matching. More...

Attributes[Orderless] = {Protected}
```

```
?? Intervall
```

```
Global`Intervall
```

```
[a_, b_] + [c_, d_] ^:= [a + c, b + d]
```

```
[a_, b_] [c_, d_] ^:=  
[a c, b d] /; c ≥ 0 && a ≥ 0
```

```
[a_, b_] [c_, d_] ^:=  
[b c, a d] /; a ≥ 0 && d ≤ 0
```

```
MakeBoxes[[a_, b_], FormatType_] ^:=  
Format[[<> ToString[a] <>, <>  
ToString[b] <>], FormatType]
```

```
Format[[a_, b_]] :=  
[<> ToString[a] <>, <> ToString[b] <>]
```

```
Intervall /:
```

```
Intervall[a_, b_] * n_Integer :=
```

```
Intervall[n a, n b] /; n > 0
```

```
?? Intervall
```

```
Global`Intervall
```

```
[a_, b_] + [c_, d_] ^:= [a + c, b + d]
```

```
[a_, b_] [c_, d_] ^:=  
[a c, b d] /; c ≥ 0 && a ≥ 0
```

```
[a_, b_] [c_, d_] ^:=  
[b c, a d] /; a ≥ 0 && d ≤ 0
```

```
Intervall /: [a_, b_] n_Integer :=  
[a n, b n] /; n > 0
```

```
MakeBoxes[[a_, b_], FormatType_] ^:=  
Format[[<> ToString[a] <>, <>  
ToString[b] <>], FormatType]
```

```
Format[[a_, b_]] :=  
[<> ToString[a] <>, <> ToString[b] <>]
```

```
Intervall[1, 3] * 7
```

```
■ [7, 21]
```