lektion2

October 22, 2025

Inhalt

1 Wiederholung I

2 Zeichenketten (engl. strings)

3 Tupel (tuple), Listen (lists), Mengen (sets), Dictionaries

4 Konstanten in Sympy

5 Symbole

6 Vereinfachungen

7 Zerlegen von Ausdrücken

8 Auswertung von Ausdrücken

1 Lektion 2

1.1 Wiederholung I

[103]:
```
1+1
```

[103]:
2

[104]:
```
2*3
```

[104]:
6

[105]:
```
1
```

[105]:
1

[106]:
```
type(1)
```

[106]:
```
int
```

[107]:
```
1.0
```

[107]:
1.0

[108]:
```
type(1.0)
```

[108]:
```
float
```

[109]:
```
1+1j
```

[109]:
```
(1+1j)
```

[110]:
```
type(1+1j)
```

[110]:
```
complex
```

1.2 Zeichenketten (engl. strings)

Zeichenkenten werden mit einfachen oder doppelten Anführungsstrich begrenzt

`" "` oder `' '`

und mit

`+`

verbunden.

[111]:
```
txt = 'Hallo'
```

[112]:
```
txt
```

[112]:
```
'Hallo'
```

[113]:
```
name = "Tom"
```

[114]:
```
txt + " " + name
```

[114]:
```
'Hallo Tom'
```

1.3 Tupel (tuple), Listen (lists), Mengen (sets), Dictionaries

Tupel werden mit ( ) begrenzt

[115]:
```
tupel = (1 , 2, 'p', 2, (3, 4))
tupel
```

[115]:
```
(1, 2, 'p', 2, (3, 4))
```

Zugriff auf die Elemente eines Tupels oder einer Liste mit [] Python fängt mit 0 an zu zählen.

[116]:
```
tupel[1]
```

[116]:
2

[ ]:
```

```

Listen werden mit [ ] Klammern begrenzt.

[117]:
```
liste = [1, 2, 'p', 2, (3, 4)]
liste
```

[117]:
```
[1, 2, 'p', 2, (3, 4)]
```

[118]:
```
liste[2]
```

[118]:
```
'p'
```

[119]:
```
liste[2] = 'q'
liste
```

[119]:
```
[1, 2, 'q', 2, (3, 4)]
```

[120]:
```
liste.append((1, 2))
liste[4]
```

[120]:
$(3,\ 4)$

Mengen werden mit { } begrenzt.

[121]:
```
menge = {2, 2, -1.0, int(1), float('1'), int('1'), 2}  # Achtung
menge
```

[121]:
$\{-1.0, 1, 2\}$

[122]:
```
float(1.0) == int(1) # '==' testet ob zwei Objekte, wenn sie ausgewertet␣
↪werden, den selben Wert haben
```

[122]:
```
True
```

[123]:
```
leereMenge = set() # nicht {}
leereMenge
```

[123]:
$\{\}$

[124]:
```
3 in menge
```

[124]:
```
False
```

Wörterbücher (Dictionaries) sind Paare von Schlüsseln (keys) und Werten (values)

[125]:
```
dictionary = {'Alice': 1234, 'Bob': 1224, 'Eve': 2346}
dictionary['Alice']
```

[125]:
1234

[126]:
```
list(dictionary.keys())
```

[126]:
```
['Alice', 'Bob', 'Eve']
```

[127]:
```
list(dictionary.values())
```

[127]:
[1234, 1224, 2346]

1.4 Konstanten in Sympy

in Sympy vordefinierte (built-in) Namen

[129]:
```
from sympy import *
init_printing()
```

[36]:
```
pi, EulerGamma, E, I, Catalan
```

[36]:
$(\pi,\ \gamma,\ e,\ i,\ G)$

[37]:
```
N(pi,10) # Kreiszahl
```

[37]:
3.141592654

[38]:
```
N(EulerGamma, 40) # Euler Gamma
```

[38]:
0.5772156649015328606065120900824024310422

[39]:
```
N(Catalan, 30) # Catalans Konstante
```

[39]:
0.915965594177219015054603514932

[40]:
```
N(E,10) # Basis des natürlichen Logarithmus
```

[40]:
2.718281828

[41]:
```
I**2 # imaginäre Einheit (aus sympy)
```

[41]:
-1

[42]:
```
1j**2 # imaginäre Einheit (aus python)
```

[42]:
```
(-1+0j)
```

[43]:
```
pi = 3
pi
```

[43]:
3

[44]:
```
from sympy import pi  #alternativ pi = S('pi')

N(pi, 4)
```

[44]:
3.142

[45]:
```
pi
```

[45]: π

1.5 Symbole

[131]:
```
m = symbols('m')
m
```

[131]: m

[47]:
```
x, y, z = symbols('x y z')
f = y*x**z+1
f
```

[47]: $x^z y + 1$

[48]:
```
xs = symbols('x:4')
xs
```

[48]: $(x_0,\ x_1,\ x_2,\ x_3)$

[144]:
```
a, b = symbols('b a') # nicht nachmachen
print('a ist', a, 'und b ist', b)
```

```
a ist b und b ist a
```

[50]:
```
xx, yy = symbols('xx yy', commutative=False)
xx * yy == yy * xx
```

[50]: False

[145]:
```
cos(m * pi)
```

[145]: $\cos(\pi m)$

[146]:
```
m = symbols('m', integer=True)
m.assumptions0
```

[146]:
```
{'algebraic': True,
 'commutative': True,
 'complex': True,
 'extended_real': True,
 'finite': True,
 'hermitian': True,
 'imaginary': False,
 'infinite': False,
 'integer': True,
 'irrational': False,
```

```
 'noninteger': False,
 'rational': True,
 'real': True,
 'transcendental': False}
```

[147]:
```
cos(m * pi)
```

[147]: $(-1)^m$

[138]:
```
b = symbols('b')
sqrt(b**2)
```

[138]: $\sqrt{b^2}$

[143]:
```
a = symbols('a', positive=True)
sqrt(a**2)
```

[143]: a

[132]:
```
n = symbols('n', positive=True, integer=True)
n.assumptions0
```

[132]:
```
{'algebraic': True,
 'commutative': True,
 'complex': True,
 'extended_negative': False,
 'extended_nonnegative': True,
 'extended_nonpositive': False,
 'extended_nonzero': True,
 'extended_positive': True,
 'extended_real': True,
 'finite': True,
 'hermitian': True,
 'imaginary': False,
 'infinite': False,
 'integer': True,
 'irrational': False,
 'negative': False,
 'noninteger': False,
 'nonnegative': True,
 'nonpositive': False,
 'nonzero': True,
 'positive': True,
 'rational': True,
 'real': True,
 'transcendental': False,
 'zero': False}
```

1.6 Vereinfachungen

[55]:
```
x, y = symbols('x y')
```

[56]:
```
f = (x - y) * (x + y)
f
```

[56]:
$$(x - y)\,(x + y)$$

[57]:
```
f.expand() # expand ist eine Methode des Objekts f
```

[57]:
$$x^2 - y^2$$

[58]:
```
expand(f)  # fast immer geht auch diese Funktionenschreibweise
```

[58]:
$$x^2 - y^2$$

[59]:
```
f.expand().factor()
```

[59]:
$$(x - y)\,(x + y)$$

[60]:
```
g = (x**2 - y**2) / (x - y)
g
```

[60]:
$$\frac{x^2 - y^2}{x - y}$$

[61]:
```
g.ratsimp()
```

[61]:
$$x + y$$

[62]:
```
g.simplify()
```

[62]:
$$x + y$$

[63]:
```
h = x*x**y
h
```

[63]:
$$xx^y$$

[64]:
```
h.powsimp()
```

[64]:
$$x^{y+1}$$

[66]:
```
f = (sin(2*x)+cos(x)) / (( (sin(2*x)**2)-cos(x)**2) * (sin(2*x)-cos(x) ))
f
```

[66]:
$$\frac{\sin(2x) + \cos(x)}{(\sin(2x) - \cos(x))\left(\sin^2(2x) - \cos^2(x)\right)}$$

[67]:
```
f.simplify()
```

[67]:
$$\frac{1}{\left(2\sin\left(x\right)-1\right)^{2}\cos^{2}\left(x\right)}$$

[68]:
```
f.trigsimp()
```

[68]:
$$\frac{1}{\left(2\sin\left(x\right)-1\right)^{2}\cos^{2}\left(x\right)}$$

[69]:
```
f.expand()
```

[69]:
$$\frac{\sin\left(2x\right)}{\sin^{3}\left(2x\right)-\sin^{2}\left(2x\right)\cos\left(x\right)-\sin\left(2x\right)\cos^{2}\left(x\right)+\cos^{3}\left(x\right)}+\frac{\cos\left(x\right)}{\sin^{3}\left(2x\right)-\sin^{2}\left(2x\right)\cos\left(x\right)-\sin\left(2x\right)\cos^{2}\left(x\right)+\cos^{3}\left(x\right)}$$

[70]:
```
f.expand(denom=True)
```

[70]:
$$\frac{\sin\left(2x\right)+\cos\left(x\right)}{\sin^{3}\left(2x\right)-\sin^{2}\left(2x\right)\cos\left(x\right)-\sin\left(2x\right)\cos^{2}\left(x\right)+\cos^{3}\left(x\right)}$$

[71]:
```
f.expand(denom=True).cancel()
```

[71]:
$$\frac{1}{\sin^{2}\left(2x\right)-2\sin\left(2x\right)\cos\left(x\right)+\cos^{2}\left(x\right)}$$

[72]:
```
f.expand(denom=True).cancel().factor()
```

[72]:
$$\frac{1}{\left(-\sin\left(2x\right)+\cos\left(x\right)\right)^{2}}$$

[73]:
```
f.expand(gibtsnicht=True)
```

[73]:
$$\frac{\sin\left(2x\right)}{\sin^{3}\left(2x\right)-\sin^{2}\left(2x\right)\cos\left(x\right)-\sin\left(2x\right)\cos^{2}\left(x\right)+\cos^{3}\left(x\right)}+\frac{\cos\left(x\right)}{\sin^{3}\left(2x\right)-\sin^{2}\left(2x\right)\cos\left(x\right)-\sin\left(2x\right)\cos^{2}\left(x\right)+\cos^{3}\left(x\right)}$$

1.7 Zerlegen von Ausdrücken

[74]:
```
f = sympify("a**2*sin(x*pi)+.09*I*10")
f
```

[74]:
$$a^{2}\sin\left(\pi x\right)+0.9i$$

[75]:
```
f.args
```

[75]:
$$\left(0.9i,\ a^{2}\sin\left(\pi x\right)\right)$$

[76]:
```
f.args[0] # Hilft nicht recht weiter
```

[76]:
$$0.9i$$

[77]:
```
f.atoms()
```

[77]:

$\{0.9, 2, i, \pi, a, x\}$

[78]:
```
f.atoms(Symbol)
```

[78]: $\{a, x\}$

[79]:
```
f.atoms(Number)
```

[79]: $\{0.9, 2\}$

[80]:
```
f.atoms(NumberSymbol)
```

[80]: $\{\pi\}$

[81]:
```
f.atoms(I)
```

[81]: $\{i\}$

[82]:
```
f
```

[82]: $a^2 \sin(\pi x) + 0.9i$

[83]:
```
f.as_coeff_add()[1][1]
```

[83]: $a^2 \sin(\pi x)$

[84]:
```
g = f.as_coeff_add()[1][1]
g
```

[84]: $a^2 \sin(\pi x)$

[85]:
```
g.as_coeff_mul()
```

[85]: $(1,\ (a^2,\ \sin(\pi x)))$

1.8 Auswertung von Ausdrücken

[86]:
```
g
```

[86]: $a^2 \sin(\pi x)$

[87]:
```
g.subs(x, 1) # Ersetzung von x durch 1 in g
```

[87]: 0

[88]:
```
h = 2 * x * y + x - y
h
```

[88]: $2xy + x - y$

[89]:
```
h.subs(x, 1).subs(y, 2)  # mehrfache Ersetzung
```

[89]: 3

[90]:
```
h.subs({x: 1, y: 2})  # mehrfache Ersetzung mit Dictionary
```

[90]: 3

[91]:
```
h.subs([(x, 1), (y, 2)])  # noch eine Alternative mit Liste von Tupeln
```

[91]: 3

[92]:
```
s = sqrt(8)
s
```

[92]: $2\sqrt{2}$

[93]:
```
s.evalf() # numerische Auswertung N(s)
```

[93]: 2.82842712474619

[133]:
```
cosc = (cos(x) - 1) / x
cosc
```

[133]: $\frac{\cos(x) - 1}{x}$

[134]:
```
cosc.evalf(subs={x: 1})
```

[134]: -0.45969769413186

[135]:
```
cosc.subs(x, 0)  # Achtung Falle
```

[135]: NaN

[97]:
```
cosc.evalf(subs={x: 0}) # Ist das richtig?
```

[97]: $\tilde{\infty}$

[98]:
```
limit(cosc, x, 0)
```

[98]: 0

[ ]:
```

```