lektion2

October 21, 2021

Table of Contents

1 Lektion 2

1.1 Wiederholung I

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

[1]:
```
2
```

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

[2]:
```
6
```

[4]:
```
1
```

[4]:
```
1
```

[5]:
```
type(1)
```

[5]:
```
int
```

[6]:
```
1.0
```

[6]:
```
1.0
```

[7]:
```
type(1.0)
```

[7]:
```
float
```

[8]:
```
1+1j
```

[8]:
```
(1+1j)
```

[9]:
```
type(1+1j)
```

[9]:
```
complex
```

1.2 Zeichenketten (engl. strings)

[11]:
```
txt = 'Hallo'
```

[12]:
```
txt
```

[12]:
```
'Hallo'
```

[14]:
```
name = 'Tom'
```

[16]:
```
txt + " " + name
```

[16]:
```
'Hallo Tom'
```

1.3 Tupel, Listen, Mengen, Dictionaries

[17]:
```
tupel = (1 , 2, 'p', 2, (3, 4))
tupel
```

[17]:
```
(1, 2, 'p', 2, (3, 4))
```

[19]:
```
tupel[1]
```

[19]:
```
2
```

[ ]:
```

```

[20]:
```
liste = [1, 2, 'p', 2, (3, 4)]
liste
```

[20]:
```
[1, 2, 'p', 2, (3, 4)]
```

[22]:
```
liste[2]
```

[22]:
```
'p'
```

[24]:
```
liste[2] = 'q'
liste
```

[24]:
```
[1, 2, 'q', 2, (3, 4)]
```

[26]:
```
liste.append((1,2))
liste[4]
```

[26]:
```
(3, 4)
```

[29]:
```
menge = {2, 2, -1.0, int(1), float('1'), int('1'), 2} # Achtung
menge
```

[29]:
```
{-1.0, 1, 2}
```

[28]:
```
float(1.0) == int(1) # '==' testet ob zwei Objekte, wenn sie ausgewertet␣
↪werden, den selben Wert haben
```

[28]:
```
True
```

[30]:
```
leereMenge = set() # nicht {}
leereMenge
```

[30]:
```
set()
```

[31]:
```
3 in menge
```

[31]:
```
False
```

[32]:
```
dictionary = {'Alice': 1234, 'Bob': 1224, 'Eve' : 2346}
dictionary['Alice']
```

[32]:
```
1234
```

[33]:
```
list(dictionary.keys())
```

[33]:
```
['Alice', 'Bob', 'Eve']
```

1.4 Wiederholung II

nochmal die Zuweisung =

[34]:
```
from sympy import *
init_printing()
```

[35]:
```
x = S('x')
y = S('y')
x, y
```

[35]: $(x,\ y)$

[36]:
```
f = x+2*y
```

[37]:
```
x = 2
```

[38]:
```
f
```

[38]: $x + 2y$

[40]:
```
f.subs(x,2)
```

[40]: $x + 2y$

[42]:
```
x
```

[42]: 2

[43]:
```
f.subs(x, 2)
```

[43]: $x + 2y$

[45]:
```
namevonx = S('x')
f.subs(S('x'), 2)
```

[45]: $2y + 2$

1.5 Konstanten in Sympy

in Sympy vordefinierte (built-in) Namen

[46]:
```
pi, EulerGamma, E, I
```

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

[48]:
```
N(pi,10) # Kreiszahl
```

[48]: 3.141592654

[49]:
```
N(EulerGamma, 40) # Euler Gamma
```

[49]: 0.5772156649015328606065120900824024310422

[50]:
```
N(Catalan, 30) # Catalans Konstante
```

[50]: 0.915965594177219015054603514932

[51]:
```
 N(E,10) # Basis des natürlichen Logarithmus
```

[51]: 2.718281828

[53]:
```
I**2 # imaginäre Einheit (aus sympy)
```

[53]: -1

[55]:
```
 1j**2 # imaginäre Einheit (aus python)
```

[55]: (-1+0j)

[57]:
```
pi = 3
pi
```

[57]: 3

[60]:
```
from sympy import pi #alternativ pi = S('pi')
N(pi,4)
```

[60]: 3.142

[ ]:
```
pi
```

1.6 Symbole

[61]:
```
m = Symbol('m')
m
```

[61]: m

[63]:
```
x, y, z = symbols('x y z')
f = y*x**z+1
f
```

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

[66]:
```
xs = symbols('x:4')
xs
```

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

[67]:
```
a, b = symbols('b a') # nicht nachmachen
print('a ist ', a, 'und b ist', b)
```

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

[71]:
```
xx, yy = symbols('xx yy' , commutative=False)
xx*yy == yy*xx
```

[71]: False

[72]:
```
cos(m*pi)
```

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

[73]:
```
m = symbols('m', integer=True)
m.assumptions0
```

[73]:
```
{'integer': True,
 'extended_real': True,
 'commutative': True,
 'rational': True,
 'real': True,
 'hermitian': True,
 'infinite': False,
 'algebraic': True,
 'complex': True,
 'transcendental': False,
 'imaginary': False,
 'noninteger': False,
 'finite': True,
 'irrational': False}
```

[74]:
```
a = Symbol('a', positive=True)
sqrt(a**2), sqrt(b**2)
```

[74]: $\left(a,\ \sqrt{a^2}\right)$

[75]:
```
n = Symbol('n', positive=True, integer=True)
n.assumptions0
```

[75]:
```
{'positive': True,
 'extended_real': True,
 'commutative': True,
 'nonpositive': False,
 'real': True,
 'extended_negative': False,
 'hermitian': True,
 'infinite': False,
 'complex': True,
 'extended_nonnegative': True,
 'nonnegative': True,
 'negative': False,
```

```
'nonzero': True,
'finite': True,
'extended_nonzero': True,
'imaginary': False,
'extended_positive': True,
'extended_nonpositive': False,
'zero': False,
'integer': True,
'rational': True,
'algebraic': True,
'transcendental': False,
'noninteger': False,
'irrational': False}
```

1.7 Vereinfachungen

[77]:
```
x, y = symbols('x y')
```

[78]:
```
f = (x-y)*(x+y)
f
```

[78]: $(x - y)(x + y)$

[79]:
```
f.expand() # expand ist eine Methode des Objekts f
```

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

[81]:
```
expand(f)  # fast immer geht auch diese Funktionenschreibweise
```

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

[82]:
```
f.expand().factor()
```

[82]: $(x - y)(x + y)$

[83]:
```
g = (x**2 -y**2)/(x-y)
g
```

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

[84]:
```
g.ratsimp()
```

[84]: $x + y$

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

[85]: $x + y$

[86]:
```
h = x*x**y
h
```

[86]:
$$xx^{y}$$

[87]:
```
h.powsimp()
```

[87]:
$$x^{y+1}$$

[88]:
```
h.powsimp().expand()
```

[88]:
$$xx^{y}$$

[89]:
```
f = (sin(2*x)+cos(x)) / (( (sin(2*x)**2)-cos(x)**2) * (sin(2*x)-cos(x) ))
f
```

[89]:
$$\frac{\sin\left(2x\right) + \cos\left(x\right)}{\left(\sin\left(2x\right) - \cos\left(x\right)\right)\left(\sin^{2}\left(2x\right) - \cos^{2}\left(x\right)\right)}$$

[90]:
```
f.simplify()
```

[90]:
$$\frac{1}{\left(2\sin\left(x\right) - 1\right)^{2}\cos^{2}\left(x\right)}$$

[91]:
```
f.trigsimp()
```

[91]:
$$\frac{1}{\left(2\sin\left(x\right) - 1\right)^{2}\cos^{2}\left(x\right)}$$

[92]:
```
f.expand()
```

[92]:
$$\frac{\sin\left(2x\right)}{\sin^{3}\left(2x\right) - \sin^{2}\left(2x\right)\cos\left(x\right) - \sin\left(2x\right)\cos^{2}\left(x\right) + \cos^{3}\left(x\right)} + \frac{\cos\left(x\right)}{\sin^{3}\left(2x\right) - \sin^{2}\left(2x\right)\cos\left(x\right) - \sin\left(2x\right)\cos^{2}\left(x\right) + \cos^{3}\left(x\right)}$$

[93]:
```
f.expand(denom=True)
```

[93]:
$$\frac{\sin\left(2x\right) + \cos\left(x\right)}{\sin^{3}\left(2x\right) - \sin^{2}\left(2x\right)\cos\left(x\right) - \sin\left(2x\right)\cos^{2}\left(x\right) + \cos^{3}\left(x\right)}$$

[94]:
```
f.expand(denom=True).cancel()
```

[94]:
$$\frac{1}{\sin^{2}\left(2x\right) - 2\sin\left(2x\right)\cos\left(x\right) + \cos^{2}\left(x\right)}$$

[95]:
```
f.expand(denom=True).cancel().factor()
```

[95]:
$$\frac{1}{\left(-\sin\left(2x\right) + \cos\left(x\right)\right)^{2}}$$

[96]:
```
f.expand(gibtsnicht=True)
```

[96]:

$$\frac{\sin(2x)}{\sin^3(2x) - \sin^2(2x)\cos(x) - \sin(2x)\cos^2(x) + \cos^3(x)} + \frac{\cos(x)}{\sin^3(2x) - \sin^2(2x)\cos(x) - \sin(2x)\cos^2(x) + \cos^3(x)}$$

1.8 Zerlegen von Ausdrücken

[98]:
```
f = sympify("a**2*sin(x*pi)+.09*I*10")
f
```

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

[100]:
```
f.args
```

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

[101]:
```
f.args[0] # Hilft nicht recht weiter
```

[101]: $0.9i$

[102]:
```
f.atoms()
```

[102]: $\{0.9, 2, i, \pi, a, x\}$

[103]:
```
f.atoms(Symbol)
```

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

[105]:
```
f.atoms(Number)
```

[105]: $\{0.9, 2\}$

[107]:
```
f.atoms(NumberSymbol)
```

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

[108]:
```
f.atoms(I)
```

[108]: $\{i\}$

[109]:
```
f
```

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

[113]:
```
f.as_coeff_add()[1][1]
```

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

[115]:
```
g = f.as_coeff_add()[1][1]
g
```

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

[116]:
```
g.as_coeff_mul()
```

[116]: $\left(1,\ \left(a^2,\ \sin\left(\pi x\right)\right)\right)$

1.9 Auswertung von Ausdrücken

[118]:
```
g
```

[118]: $a^2 \sin\left(\pi x\right)$

[120]:
```
g.subs(x, 1) # Ersetzung von x durch 1 in g
```

[120]: 0

[121]:
```
h = 2*x*y+x-y
h
```

[121]: $2xy + x - y$

[123]:
```
h.subs(x,1).subs(y,2) # mehrfache Ersetzung
```

[123]: 3

[124]:
```
h.subs({x:1, y:2}) # mehrfache Ersetzung mit Dictionary
```

[124]: 3

[125]:
```
h.subs([(x, 1), (y, 2)]) # noch eine Alternative mit Liste von Tupeln
```

[125]: 3

[126]:
```
s = sqrt(8)
s
```

[126]: $2\sqrt{2}$

[127]:
```
s.evalf() # numerische Auswertung N(s)
```

[127]: 2.82842712474619

[128]:
```
cosc = (cos(x)-1)/x
cosc
```

[128]: $\dfrac{\cos\left(x\right) - 1}{x}$

[131]:
```
cosc.evalf(subs={x: 1})
```

[131]: -0.45969769413186

[132]:
```
cosc.subs(x, 0) # Achtung Falle
```

[132]:
NaN

[133]:
```
cosc.evalf(subs={x:0})
```

[133]:
0