lektion3

October 28, 2021

Inhalt

1 Lektion 3

1.1 Wiederholung

[1]:
```
from sympy import *
init_printing()
```

[2]:
```
x, n = symbols('x n')
```

[3]:
```
cosc = (cos(x)-1)/x
cosc
```

[3]:
$$\frac{\cos\left(x\right) - 1}{x}$$

[4]:
```
cosc.subs(x, 0) # Achtung Falle
```

[4]:
NaN

[5]:
```
cosc.evalf(subs={x:0}) # das ist richtig
```

[5]:
0

1.2 Grenzwerte

[6]:
```
cosc.limit(x, 0)
```

[6]:
0

[7]:
```
limit(cosc, x, 0)
```

[7]:
0

[8]:
```
Limit(cosc, x, 0) # träger Operator
```

[8]:
$$\lim_{x \to 0^+} \left(\frac{\cos(x) - 1}{x} \right)$$

[9]:
```
L = Limit(1/x, x, 0)
L
```

[9]:
$$\lim_{x \to 0^+} \frac{1}{x}$$

[10]:
```
L.doit()
```

[10]:
∞

[11]:
```
Limit(1/x, x, 0, '-' ).doit()
```

[11]:
$-\infty$

[12]:
```
b = factorial(n)/sqrt(n)*(E/n)**n
b
```

[12]:
$$\frac{\left(\frac{e}{n}\right)^n n!}{\sqrt{n}}$$

[13]:
```
L = Limit(b, n, oo)
L
```

[13]:
$$\lim_{n \to \infty} \left(\frac{\left(\frac{e}{n}\right)^n n!}{\sqrt{n}} \right)$$

[14]:
```
L.doit()
```

[14]:
$\sqrt{2}\sqrt{\pi}$

[15]:
```
bb = b.evalf(subs={n: 10000})
```

[16]:
```
N(bb - sqrt(2*pi))
```

[16]:
$$2.08886559842248 \cdot 10^{-5}$$

1.3 Summen und Reihen

[17]:
```
m = S('m')
s = Sum(m, (m, 0, n))
s
```

[17]:
$$\sum_{m=0}^{n} m$$

[18]:
```
s.doit()
```

[18]:
$$\frac{n^2}{2} + \frac{n}{2}$$

[19]:
```
s.doit().factor()
```

[19]:
$$\frac{n\,(n+1)}{2}$$

[20]:
```
g = Sum(x**m, (m, 0, n))
g
```

[20]:
$$\sum_{m=0}^{n} x^m$$

[21]:
```
g.doit()
```

[21]:
$$\begin{cases} n+1 & \text{for } x = 1 \\ \frac{1-x^{n+1}}{1-x} & \text{otherwise} \end{cases}$$

[22]:
```
g = Sum(x**m, (m, 0, oo)) # unendlich oo "zwei kleine o"
g
```

[22]:
$$\sum_{m=0}^{\infty} x^m$$

[23]:
```
g.doit()
```

[23]:
$$\begin{cases} \frac{1}{1-x} & \text{for } |x| < 1 \\ \sum_{m=0}^{\infty} x^m & \text{otherwise} \end{cases}$$

[24]:
```
g.doit().args
```

[24]:
$$\left(\left(\frac{1}{1-x},\ |x|<1\right),\ \left(\sum_{m=0}^{\infty} x^m,\ \text{True}\right)\right)$$

[25]:
```
g.doit().args[0]
```

[25]:
$$\left(\frac{1}{1-x},\ |x|<1\right)$$

[26]:
```
g.doit().args[0][0]
```

[26]:
$$\frac{1}{1-x}$$

[27]:
```
a = Sum((-1)**(m+1)/m,(m,1,oo))
a
```

[27]:
$$\sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m}$$

[28]:
```
a.doit()
```

[28]:
$$\log(2)$$

[29]:
```
s = Sum(1/m**2,(m,1,oo))
s
```

[29]:
$$\sum_{m=1}^{\infty} \frac{1}{m^2}$$

[30]:
```
s.doit()
```

[30]:
$$\frac{\pi^2}{6}$$

1.4 Ableitungen

[31]:
```
f = x**n
```

[32]:
```
f.diff(x)
```

[32]:
$$\frac{nx^n}{x}$$

[33]:
```
f.diff(x).powsimp()
```

[33]:
$$nx^{n-1}$$

[34]:
```
f.diff(x,x,x)
```

[34]:

$$\frac{nx^n\left(n^2-3n+2\right)}{x^3}$$

[35]:
```
f.diff(x,x,x).powsimp().factor()
```

[35]:
$$nx^{n-3}\left(n-2\right)\left(n-1\right)$$

[36]:
```
f.diff(x,3).powsimp().factor()
```

[36]:
$$nx^{n-3}\left(n-2\right)\left(n-1\right)$$

[37]:
```
f.diff(x,0)
```

[37]:
$$x^n$$

[38]:
```
f.diff()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_16312/968738442.py in <module>
----> 1 f.diff()

/local/home/schaedle/miniconda3/lib/python3.9/site-packages/sympy/core/expr.py␣
 ↪in diff(self, *symbols, **assumptions)
   3500     def diff(self, *symbols, **assumptions):
   3501         assumptions.setdefault("evaluate", True)
-> 3502         return _derivative_dispatch(self, *symbols, **assumptions)
   3503
   3504 ␣
 ↪############################################################################

/local/home/schaedle/miniconda3/lib/python3.9/site-packages/sympy/core/function
 ↪py in _derivative_dispatch(expr, *variables, **kwargs)
   1945         from sympy.tensor.array.array_derivatives import ArrayDerivativ
   1946         return ArrayDerivative(expr, *variables, **kwargs)
-> 1947     return Derivative(expr, *variables, **kwargs)
   1948
   1949

/local/home/schaedle/miniconda3/lib/python3.9/site-packages/sympy/core/function
 ↪py in __new__(cls, expr, *variables, **kwargs)
   1294                             to differentiate %s''' % expr))
   1295                 else:
-> 1296                     raise ValueError(filldedent('''
   1297                         Since there is more than one variable in the
   1298                         expression, the variable(s) of differentiation

ValueError:
```

```
Since there is more than one variable in the expression, the
variable(s) of differentiation must be supplied to differentiate x**n
```

[39]:
```
g = exp(x) # das geht, das in dem Ausdruck exp(x) nur ein Symbol vorkommt
g.diff()
```

[39]: e^{x}

[40]:
```
y = symbols('y')
```

[41]:
```
g = exp(x*y)
```

[42]:
```
g.diff(x, x, y, y, y)
```

[42]: $x\left(x^{2}y^{2}+6xy+6\right)e^{xy}$

[43]:
```
g.diff(x, 2, y, 3)
```

[43]: $x\left(x^{2}y^{2}+6xy+6\right)e^{xy}$

1.5 Grenzwert mit der Regel von l'Hospital

[44]:
```
a = (cos(pi*x)-1+pi*x/2)/x
```

[45]:
```
na = numer(a)    # numer gibt es nicht als methode
da = denom(a)
```

[46]:
```
na.diff(x)
```

[46]: $-\pi\sin\left(\pi x\right)+\frac{\pi}{2}$

[47]:
```
n1 = na.diff(x).subs(x, 0) # hier kann man gefahrlos substituieren
n1
```

[47]: $\frac{\pi}{2}$

[48]:
```
d1 = da.diff(x).subs(x, 0)
d1
```

[48]: 1

[49]:
```
l = n1/d1
l
```

[49]: $\frac{\pi}{2}$

[50]:
```
l ==  a.limit(x, 0)
```

[50]:
```
True
```

1.6 Unbestimmte Integrale

[51]:
```
f = x**n
f
```

[51]: x^n

[52]:
```
I = Integral(f, x)  # Achtung jetzt haben wird die imaginäre Einheite␣
↪überschrieben
I
```

[52]: $\int x^n \, dx$

[53]:
```
F = I.doit()
F
```

[53]: $\begin{cases} \frac{x^{n+1}}{n+1} & \text{for } n \neq -1 \\ \log\left(x\right) & \text{otherwise} \end{cases}$

[54]:
```
F.diff(x).simplify()
```

[54]: $\begin{cases} x^{n} & \text{for } n \neq -1 \\ \frac{1}{x} & \text{otherwise} \end{cases}$

[55]:
```
g = 1/(1+x**4)
I2 = Integral(g, x)
I2
```

[55]: $\int \frac{1}{x^4 + 1} \, dx$

[56]:
```
G = I2.doit()
G
```

[56]: $-\frac{\sqrt{2}\log\left(x^2 - \sqrt{2}x + 1\right)}{8} + \frac{\sqrt{2}\log\left(x^2 + \sqrt{2}x + 1\right)}{8} + \frac{\sqrt{2}\operatorname{atan}\left(\sqrt{2}x - 1\right)}{4} + \frac{\sqrt{2}\operatorname{atan}\left(\sqrt{2}x + 1\right)}{4}$

[58]:
```
G.diff(x) == g
```

[58]:
```
False
```

[59]:
```
G.diff(x)
```

[59]: $-\frac{\sqrt{2}\left(2x - \sqrt{2}\right)}{8\left(x^2 - \sqrt{2}x + 1\right)} + \frac{\sqrt{2}\left(2x + \sqrt{2}\right)}{8\left(x^2 + \sqrt{2}x + 1\right)} + \frac{1}{2\left(\left(\sqrt{2}x + 1\right)^2 + 1\right)} + \frac{1}{2\left(\left(\sqrt{2}x - 1\right)^2 + 1\right)}$

[60]:
```
G.diff(x).ratsimp()
```

[60]:
$$\frac{1}{x^4+1}$$

Um die Gleichheit zweier Ausdrücke zur testen vereinfacht man besser die Differenz, statt mit == zu testen.

[61]:
```
simplify(G.diff(x) - g)
```

[61]:
$$0$$

1.7 Bestimmte Integrale

[62]:
```
Integral(x*(x-n), (x, 0, n))
```

[62]:
$$\int_0^n x\left(-n+x\right)\,dx$$

[64]:
```
Integral(x*(x-n), (x, 0, n)).doit()
```

[64]:
$$-\frac{n^3}{6}$$

[63]:
```
I3 = Integral(g, (x, -oo, oo))
I3
```

[63]:
$$\int_{-\infty}^{\infty} \frac{1}{x^4+1}\,dx$$

[65]:
```
I3.doit()
```

[65]:
$$\frac{\sqrt{2}\pi}{2}$$

[66]:
```
G.limit(x, oo) - G.limit(x, -oo)
```

[66]:
$$\frac{\sqrt{2}\pi}{2}$$

1.8 For Schleifen

[67]:
```
list(range(10))
```

[67]:
$$[0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9]$$

[68]:
```
for j in range(10):
    print('innen\t', j)
print('aussen\t', j+100)
```

```
innen    0
innen    1
innen    2
innen    3
innen    4
innen    5
innen    6
innen    7
innen    8
innen    9
aussen   109
```

[69]:
```
from sympy import Rational
```

Blöcke werden durch Einrückung/Ausrückung festgelegt

[71]:
```
for p in range(-3, 3):
    for q in range(1, 4):
        print(f"p = {p}, q = {q}")
        """ f-String: für {x} wird der Wert von x eingesetzt.
                      Die geschweiften Klammern sind ein Platzhalten """
        display(Rational(p, q))
```

```
p = -3, q = 1
```

-3

```
p = -3, q = 2
```

$-\frac{3}{2}$

```
p = -3, q = 3
```

-1

```
p = -2, q = 1
```

-2

```
p = -2, q = 2
```

-1

```
p = -2, q = 3
```

$-\frac{2}{3}$

```
p = -1, q = 1
```

-1

```
p = -1, q = 2
```

$$-\frac{1}{2}$$

```
p = -1, q = 3
```

$$-\frac{1}{3}$$

```
p = 0, q = 1
```

$$0$$

```
p = 0, q = 2
```

$$0$$

```
p = 0, q = 3
```

$$0$$

```
p = 1, q = 1
```

$$1$$

```
p = 1, q = 2
```

$$\frac{1}{2}$$

```
p = 1, q = 3
```

$$\frac{1}{3}$$

```
p = 2, q = 1
```

$$2$$

```
p = 2, q = 2
```

$$1$$

```
p = 2, q = 3
```

$$\frac{2}{3}$$

[72]:
```
summe = 0
N = 20
for n in range(1, N):
    summe += Rational(1, n)
summe
```

[72]:
$$\frac{275295799}{77597520}$$

1.9 Vergleiche

[74]:
```
1 == 0  # gleich
```

[74]:
```
False
```

[75]:
```
1 != 0 # ungleich
```

[75]:
```
True
```

[76]:
```
1 <= 2 # kleinergleich
```

[76]:
```
True
```

Weiter gibt es < (kleiner) und > (größer) und >= (größergleich)

1.10 Logische Verknüpfungen

[77]:
```
True, False #boolsche Variablen
```

[77]:
```
(True, False)
```

[79]:
```
not True # Negation
```

[79]:
```
False
```

zweistellige logische Verknüpfungen sind **and** für "und" und **or** für "oder"

[80]:
```
for a in [True, False]:
    for b in [True, False]:
        print("{0:5} and {1:5} ergibt {2}".format(str(a), str(b), a and b))
        """ wandelt boolschen Variablen in einen string um str() und verwendet␣
↪5 Stellen zur Anzeige.
        Wieder sind die geschweiften Klammern ein Platzhalter {0} wird durch␣
↪das erste Argument, {1}
        durch das zweite, usw ersetzt """
```

```
True  and True  ergibt True
True  and False ergibt False
False and True  ergibt False
False and False ergibt False
```

[81]:
```
for a in [True, False]:
    for b in [True, False]:
        print(f"{a} or {b} == {a or b}")
```

```
True or True == True
True or False == True
```

```
False or True == True
False or False == False
```

alternativ kann man das bitwise & für “und” und | für “oder” verwenden

[82]:
```
for a in [True, False]:
    for b in [True, False]:
        print("{0:5} & {1:5} ergibt {2}".format(str(a),str(b),a&b))

print("\n") # \n 'newline' macht einen Zeilenumbruch in der Ausgabe
for a in [True, False]:
    for b in [True, False]:
        print(f"{str(a):5} | {str(b):5} == {a|b}")
```

```
True  & True  ergibt True
True  & False ergibt False
False & True  ergibt False
False & False ergibt False


True  | True  == True
True  | False == True
False | True  == True
False | False == False
```

[83]:
```
1<2<1 # äquivalent zu  (1 < 2) und (2 < 1)
```

[83]:
```
False
```

1.11 Bedingte Anweisung und Verzweigungen

[84]:
```
a = 42
if 1<a<100:
    print('richitg:', a)
```

```
richitg: 42
```

[85]:
```
a = 200
if 1<a<100:
    print('richitg:', a)
else:
    print('zu groß oder zu klein ')
```

```
zu groß oder zu klein
```

[86]:
```
a = 0
if 1<a<100:
```

```
    print('richitg:', a)
elif a>=100:
    print('zu groß')
else:
    print('zu klein')
```

```
zu klein
```

1.11.1 Beispiel

Finde das kleinste m so, dass

$$\sum_{j=1}^{m} \frac{1}{j} \geq 5$$

Achtung! Das war in der Vorlesung noch falsch

[91]:
```
summe = 0
for m in range(1, 1000000):
    summe += Rational(1,m)
    if summe >= 5:
        break

summe.evalf(), m
```

[91]:
```
(5.00206827268017, 83)
```

break bricht eine Schleife ab (bei verschachtelten Schleifen die innerste)

1.12 While Schleife

[93]:
```
summe = 0
m = 0
while summe < 5:
    m += 1
    summe += Rational(1, m)

summe.evalf(), m
```

[93]:
```
(5.00206827268017, 83)
```

[ ]:
```
```