05d_SymPy_Miniuebungen_Loesungen

0.1 Mini-Aufgaben zur Überprüfung des Verständnis: Grundlagen von SymPy

0.1.1 SymPy als Hilfsmittel für Beweise

In dieser Miniübung nutzen wir SymPy, um auf verschiedene Arten die Gültigkeit eines Additionstheorems für Sinus und Cosinus zu beweisen. Realisieren Sie als ersten Schritt die folgenden Ausdrücke mit SymPy:

$$f(x,y) = \sin(x+y)$$
$$g(x,y) = \cos(x)\sin(y) + \sin(x)\cos(y)$$

[3]:
```
import sympy as sp

sp.init_printing(use_unicode=True)

x,y = sp.symbols('x y')
f_sp = sp.sin(x+y)
display(f_sp)
g_sp = sp.cos(x)*sp.sin(y)+sp.sin(x)*sp.cos(y)
display(g_sp)
```

$\sin(x+y)$

$\sin(x)\cos(y) + \sin(y)\cos(x)$

0.1.2 Schritt 2

Versuchen Sie, die Gleichheit der beiden Ausdrücke mit SymPy zu zeigen.

[8]:
```
import sympy as sp

sp.init_printing(use_unicode=True)

x,y = sp.symbols('x y')
f_sp = sp.sin(x+y)
g_sp = sp.cos(x)*sp.sin(y)+sp.sin(x)*sp.cos(y)

print("Direkter Vergleich     :", f_sp==g_sp)
print("Vergleich nach simplify:", sp.simplify(f_sp)==sp.simplify(g_sp))
```

```
print("Display nach simplify:")
display(sp.simplify(f_sp))
display(sp.simplify(g_sp))
```

```
Direkter Vergleich     : False
Vergleich nach simplify: True
Display nach simplify:
```

$$\sin(x+y)$$

$$\sin(x+y)$$

0.1.3 Schritt 3

Bestätigen Sie das Ergebnis durch eine numerische Evaluierung in $x = 0$ und $y = 0$.

[1]:
```
import sympy as sp

sp.init_printing(use_unicode=True)

x,y = sp.symbols('x y')
f_sp = sp.sin(x+y)
g_sp = sp.cos(x)*sp.sin(y)+sp.sin(x)*sp.cos(y)

eval_dict = {x:0,y:0}
print("f(0,0) und g(0,0) numerisch:", f_sp.evalf(subs=eval_dict), " und ", g_sp.
 ↪evalf(subs=eval_dict))
```

```
f(0,0) und g(0,0) numerisch: 0  und  0
```

0.1.4 Schritt 4

Bestätigen Sie das Ergebnis durch die Verwendung der SymPy-Approximation equals().

[15]:
```
import sympy as sp

sp.init_printing(use_unicode=True)

x,y = sp.symbols('x y')
f_sp = sp.sin(x+y)
g_sp = sp.cos(x)*sp.sin(y)+sp.sin(x)*sp.cos(y)

print(f_sp.equals(g_sp))
```

```
True
```

0.1.5 Schritt 5

Bestätigen Sie das Ergebnis durch 100 Vergleiche zwischen der Evaluierung der SymPy-Varianten und der entsprechenden NumPy-Varianten, im Intervall $[0, 2\pi]$

[31]:
```
import sympy as sp
import numpy as np

sp.init_printing(use_unicode=True)

# Das ist Standard-SymPy
x,y = sp.symbols('x y')
f_sp = sp.sin(x+y)
g_sp = sp.cos(x)*sp.sin(y)+sp.sin(x)*sp.cos(y)

# Das ist Standard-NumPy
comparison_points = np.array(np.random.random(100))

# Die explizite Konvertierung zu float ist nervig, der Rest ist klar
f_vals_sp = np.array([float(f_sp.evalf(subs={x:i,y:i})) for i in␣
 ↪comparison_points])
g_vals_sp = np.array([float(g_sp.evalf(subs={x:i,y:i})) for i in␣
 ↪comparison_points])

# Das ist sowieso klar
f_vals_np = np.sin(comparison_points+comparison_points)
g_vals_np = np.cos(comparison_points)*np.sin(comparison_points) + np.
 ↪sin(comparison_points)*np.cos(comparison_points)

# Hierfür ist die explizite Konvertierung zu float weiter oben nötig
res_f = np.linalg.norm (f_vals_sp - f_vals_np)
res_g = np.linalg.norm (g_vals_sp - g_vals_np)

# Juchu
print(res_f - res_g)
```

```
-4.38800252352545e-16
```

0.1.6 Wiederholen Sie diese Übung für $\sin(x)^2 + \cos(x)^2 = 1$

[32]:
```
import sympy as sp

sp.init_printing(use_unicode=True)
x = sp.symbols('x')
expr = sp.sin(x)**2+sp.cos(x)**2
display(expr)

# ...
```

$$\sin^2(x) + \cos^2(x)$$

0.2 Impressum

0.2.1 Programmierkurs Python, Dominik Göddeke https://www.ians.uni-stuttgart.de, Universität Stuttgart

Version vom April 2023

Veröffentlicht auf https://zoerr.de, (alle Rechte am Logo vorbehalten)

Gefördert durch die Stiftung Innovation in der Hochschullehre. (alle Rechte am Logo vorbehalten)

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie.

[ ]: