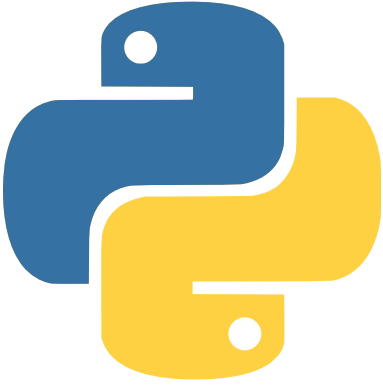
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

SymPy: Einleitung, Ausdrücke

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

SymPy:
Einleitung, Ausdrücke

Einführung in die symbolische Rechnung

- **SymPy**: Python-Paket für symbolische Berechnungen und Computeralgebra

Einführung in die symbolische Rechnung

- **SymPy**: Python-Paket für symbolische Berechnungen und Computeralgebra
- `Symbolische Rechnung`: stark vereinfacht „Gegenteil“ zur numerisch-approximativen Arbeit mit Gleitkomma-Zahlen

Einführung in die symbolische Rechnung

- **SymPy**: Python-Paket für symbolische Berechnungen und Computeralgebra
- `Symbolische Rechnung`: stark vereinfacht „Gegenteil“ zur numerisch-approximativen Arbeit mit Gleitkomma-Zahlen
  - Computer arbeitet mathematisch exakt, folgt den Grundregeln der Mathematik

Einführung in die symbolische Rechnung

- **SymPy**: Python-Paket für symbolische Berechnungen und Computeralgebra
- `Symbolische Rechnung`: stark vereinfacht „Gegenteil“ zur numerisch-approximativen Arbeit mit Gleitkomma-Zahlen
  - Computer arbeitet mathematisch exakt, folgt den Grundregeln der Mathematik
  - Aber: numerische Approximation fast immer schneller

Einführung in die symbolische Rechnung

- **SymPy**: Python-Paket für symbolische Berechnungen und Computeralgebra
- `Symbolische Rechnung`: stark vereinfacht „Gegenteil“ zur numerisch-approximativen Arbeit mit Gleitkomma-Zahlen
  - Computer arbeitet mathematisch exakt, folgt den Grundregeln der Mathematik
  - Aber: numerische Approximation fast immer schneller
  - Beweisbar: nicht alles berechenbar in der Computeralgebra

Einführung in die symbolische Rechnung

- **SymPy**: Python-Paket für symbolische Berechnungen und Computeralgebra
- `Symbolische Rechnung`: stark vereinfacht „Gegenteil“ zur numerisch-approximativen Arbeit mit Gleitkomma-Zahlen
  - Computer arbeitet mathematisch exakt, folgt den Grundregeln der Mathematik
  - Aber: numerische Approximation fast immer schneller
  - Beweisbar: nicht alles berechenbar in der Computeralgebra
- Moral: symbolische und numerische Mathematik gleichberechtigt

Umfang des Pakets SymPy

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie
  - Grenzwertbildung, Differentiation, Integration und Taylorentwicklungen

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie
  - Grenzwertbildung, Differentiation, Integration und Taylorentwicklungen
  - Gleichungssystemlöser für Polynomgleichungen, arithmetische Ausdrücke, Differential- und Integralgleichungen

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie
  - Grenzwertbildung, Differentiation, Integration und Taylorentwicklungen
  - Gleichungssystemlöser für Polynomgleichungen, arithmetische Ausdrücke, Differential- und Integralgleichungen
  - Kombinatorik

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie
  - Grenzwertbildung, Differentiation, Integration und Taylorentwicklungen
  - Gleichungssystemlöser für Polynomgleichungen, arithmetische Ausdrücke, Differential- und Integralgleichungen
  - Kombinatorik
  - Matrixrechnung

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie
  - Grenzwertbildung, Differentiation, Integration und Taylorentwicklungen
  - Gleichungssystemlöser für Polynomgleichungen, arithmetische Ausdrücke, Differential- und Integralgleichungen
  - Kombinatorik
  - Matrixrechnung
  - Elementare Geometrie wie Schnittpunktprobleme, Tangentenberechnung

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie
  - Grenzwertbildung, Differentiation, Integration und Taylorentwicklungen
  - Gleichungssystemlöser für Polynomgleichungen, arithmetische Ausdrücke, Differential- und Integralgleichungen
  - Kombinatorik
  - Matrixrechnung
  - Elementare Geometrie wie Schnittpunktprobleme, Tangentenberechnung
  - verschiedenste Kryptographie-Verfahren

Umfang des Pakets SymPy

- Vollständige Übersicht auf der Projektwebseite
  - Termvereinfachungen bei beinahe beliebigen arithmetischen Ausdrücken
  - Polynomrechnung und Trigonometrie
  - Grenzwertbildung, Differentiation, Integration und Taylorentwicklungen
  - Gleichungssystemlöser für Polynomgleichungen, arithmetische Ausdrücke, Differential- und Integralgleichungen
  - Kombinatorik
  - Matrixrechnung
  - Elementare Geometrie wie Schnittpunktprobleme, Tangentenberechnung
  - verschiedenste Kryptographie-Verfahren
- Bestandteil von Anaconda, alternativ per `pip`

Symbole und arithmetische Ausdrücke

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**
- Problem: Mischung bspw. mit NumPy, lösen wir unterwegs

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**
- Problem: Mischung bspw. mit NumPy, lösen wir unterwegs
- Konvention 2: „Schönschrift-Modus“

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**
- Problem: Mischung bspw. mit NumPy, lösen wir unterwegs
- Konvention 2: „Schönschrift-Modus“
  - `init_printing(use_unicode=True)`

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**
- Problem: Mischung bspw. mit NumPy, lösen wir unterwegs
- Konvention 2: „Schönschrift-Modus“
  - `init_printing(use_unicode=True)`
- Wichtigster Schritt: Deklaration von Variablen als sogenannte **Symbole**

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**
- Problem: Mischung bspw. mit NumPy, lösen wir unterwegs
- Konvention 2: „Schönschrift-Modus“
  - `init_printing(use_unicode=True)`
- Wichtigster Schritt: Deklaration von Variablen als sogenannte **Symbole**
- Dann: Arbeit fast wie mit Papier und Bleistift

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**
- Problem: Mischung bspw. mit NumPy, lösen wir unterwegs
- Konvention 2: „Schönschrift-Modus“
  - `init_printing(use_unicode=True)`
- Wichtigster Schritt: Deklaration von Variablen als sogenannte **Symbole**
- Dann: Arbeit fast wie mit Papier und Bleistift
- Praktisch: **elementare Vereinfachungen** automatisch, wie Elimination additiver Nullen, multiplikativer Einsen, Kürzungen oder Zusammenfassen von Zahlwerten

Symbole und arithmetische Ausdrücke

- Konvention, zunächst: `from sympy import *`
- Erspart Sucharbeit, wo bspw. `sqrt()` realisiert ist
- Annahme: Nutzung als Taschenrechner, **Computeralgebra-System**
- Problem: Mischung bspw. mit NumPy, lösen wir unterwegs
- Konvention 2: „Schönschrift-Modus“
  - `init_printing(use_unicode=True)`
- Wichtigster Schritt: Deklaration von Variablen als sogenannte **Symbole**
- Dann: Arbeit fast wie mit Papier und Bleistift
- Praktisch: **elementare Vereinfachungen** automatisch, wie Elimination additiver Nullen, multiplikativer Einsen, Kürzungen oder Zusammenfassen von Zahlwerten
- `display(expression)` für Bildschirmausgaben

Codebeispiele

Impressum, Danksagung und Quellen

Stiftung Innovation in der Hochschullehre

Gefördert durch die Stiftung Innovation in der Hochschullehre im Rahmen des Projekts digit@L, https://stiftung-hochschullehre.de

SimTech

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

---

Autor: Dominik Göddeke, IANS, Universität Stuttgart

ZOERR

Veröffentlicht auf dem Zentralen OER Repositorium Baden-Württemberg, https://www.zoerr.de