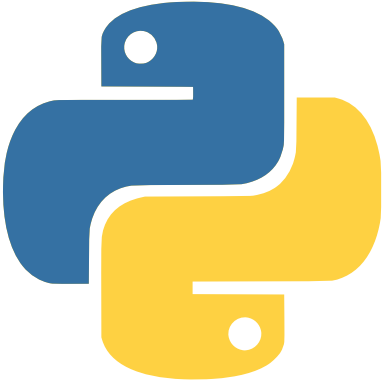
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

NumPy: Rechnen, Indizieren und Dateien

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

NumPy:
Rechnen, Indizieren und Dateien

Rechnen, Indizieren und Dateien

- Großer Vorteil von NumPy: Standardoperationen mit Matrizen und Vektoren bereits **sehr effizient** implementiert

Rechnen, Indizieren und Dateien

- Großer Vorteil von NumPy: Standardoperationen mit Matrizen und Vektoren bereits **sehr effizient** implementiert
- Unter der Haube auf eine hocheffiziente native Bibliothek durchgereicht

Rechnen, Indizieren und Dateien

- Großer Vorteil von NumPy: Standardoperationen mit Matrizen und Vektoren bereits **sehr effizient** implementiert
- Unter der Haube auf eine hocheffiziente native Bibliothek durchgereicht
  - **BLAS**, Basic Linear Algebra Subprograms

Rechnen, Indizieren und Dateien

- Großer Vorteil von NumPy: Standardoperationen mit Matrizen und Vektoren bereits **sehr effizient** implementiert
- Unter der Haube auf eine hocheffiziente native Bibliothek durchgereicht
  - **BLAS**, Basic Linear Algebra Subprograms
  - Viel schneller als jede interpretierte Sprache wie Python

Rechnen, Indizieren und Dateien

- Großer Vorteil von NumPy: Standardoperationen mit Matrizen und Vektoren bereits **sehr effizient** implementiert
- Unter der Haube auf eine hocheffiziente native Bibliothek durchgereicht
  - **BLAS**, Basic Linear Algebra Subprograms
  - Viel schneller als jede interpretierte Sprache wie Python
- **In diesem Abschnitt**: Demonstration dieses Vorteils

Rechnen, Indizieren und Dateien

- Großer Vorteil von NumPy: Standardoperationen mit Matrizen und Vektoren bereits **sehr effizient** implementiert
- Unter der Haube auf eine hocheffiziente native Bibliothek durchgereicht
  - **BLAS**, Basic Linear Algebra Subprograms
  - Viel schneller als jede interpretierte Sprache wie Python
- **In diesem Abschnitt**: Demonstration dieses Vorteils
- Und Details zur Indizierung von NumPy-Vektoren und -Matrizen

Rechnen, Indizieren und Dateien

- Großer Vorteil von NumPy: Standardoperationen mit Matrizen und Vektoren bereits **sehr effizient** implementiert
- Unter der Haube auf eine hocheffiziente native Bibliothek durchgereicht
  - **BLAS**, Basic Linear Algebra Subprograms
  - Viel schneller als jede interpretierte Sprache wie Python
- **In diesem Abschnitt**: Demonstration dieses Vorteils
- Und Details zur Indizierung von NumPy-Vektoren und -Matrizen
- Sowie: Laden und Speichern von `ndarrays` als Dateien

Elementweise Operationen

Elementweise Operationen

- **Standardoperationen** wie Addition, Subtraktion auf Vektoren, Matrizen und allgemeinen `ndarrays`: **immer elementweise**

Elementweise Operationen

- **Standardoperationen** wie Addition, Subtraktion auf Vektoren, Matrizen und allgemeinen `ndarrays`: **immer elementweise**
- `a*b` ist keine Matrixmultiplikation

Elementweise Operationen

- **Standardoperationen** wie Addition, Subtraktion auf Vektoren, Matrizen und allgemeinen `ndarrays`: **immer elementweise**
- `a*b` ist keine Matrixmultiplikation
- Sondern eine elementweise Multiplikation

Codebeispiele

Skalarprodukte, Matrix-Vektor und Matrix-Matrix Multiplikationen

Skalarprodukte etc.

- `dot()` Funktion eines `ndarray`

Skalarprodukte etc.

- `dot()` Funktion eines `ndarray`
- Entspricht mathematischer Definition

Skalarprodukte etc.

- `dot()` Funktion eines `ndarray`
- Entspricht mathematischer Definition
  - $x^\top y$ für Vektoren

Skalarprodukte etc.

- `dot()` Funktion eines `ndarray`
- Entspricht mathematischer Definition
  - $x^\top y$ für Vektoren
  - „Zeilenvektor dot Spaltenvektor“ für Matrix-Vektor- und Matrix-Matrix-Produkte

Skalarprodukte etc.

- `dot()` Funktion eines `ndarray`
- Entspricht mathematischer Definition
  - $x^\top y$ für Vektoren
  - „Zeilenvektor dot Spaltenvektor“ für Matrix-Vektor- und Matrix-Matrix-Produkte
- Ab Python 3.5: `@` Operator

Codebeispiele

Indizierung von Vektoren und Matrizen

Indizierung von Vektoren und Matrizen

- Zugriff auf einzelne Einträge eines Vektors und einer Matrix: so **wie von Listen gewohnt**

Indizierung von Vektoren und Matrizen

- Zugriff auf einzelne Einträge eines Vektors und einer Matrix: so **wie von Listen gewohnt**
- Indizierung bei Null

Indizierung von Vektoren und Matrizen

- Zugriff auf einzelne Einträge eines Vektors und einer Matrix: so **wie von Listen gewohnt**
- Indizierung bei Null
- Doppelpunkt-Operator behält Bedeutung

Codebeispiele

Vektorisierte Berechnung

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Aufgabe: Berechnung von $(f(x_1), f(x_2), \ldots, f(x_n))$

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Aufgabe: Berechnung von $(f(x_1), f(x_2), \ldots, f(x_n))$
- Möglich per `for` Schleife

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Aufgabe: Berechnung von $(f(x_1), f(x_2), \ldots, f(x_n))$
- Möglich per `for` Schleife
- Für hinreichend große n nicht sehr effizient

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Aufgabe: Berechnung von $(f(x_1), f(x_2), \ldots, f(x_n))$
- Möglich per `for` Schleife
- Für hinreichend große n nicht sehr effizient
- Deshalb: bereitgestellte NumPy Funktionen

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Aufgabe: Berechnung von $(f(x_1), f(x_2), \ldots, f(x_n))$
- Möglich per `for` Schleife
- Für hinreichend große n nicht sehr effizient
- Deshalb: bereitgestellte NumPy Funktionen
  - Grundrechenarten, Ersetzung der Bibliothek `math`

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Aufgabe: Berechnung von $(f(x_1), f(x_2), \ldots, f(x_n))$
- Möglich per `for` Schleife
- Für hinreichend große n nicht sehr effizient
- Deshalb: bereitgestellte NumPy Funktionen
  - Grundrechenarten, Ersetzung der Bibliothek `math`
  - NumPy-Funktionen kompatibel mit `math`

Vektorisierte Berechnung

- Häufiges Szenario: Anwendung einer bestimmten Operation **elementweise** auf einen Vektor
  - Gegeben: Vektor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Aufgabe: Berechnung von $(f(x_1), f(x_2), \ldots, f(x_n))$
- Möglich per `for` Schleife
- Für hinreichend große n nicht sehr effizient
- Deshalb: bereitgestellte NumPy Funktionen
  - Grundrechenarten, Ersetzung der Bibliothek `math`
  - NumPy-Funktionen kompatibel mit `math`
  - Gleiche Funktionalität, aber deutlich schneller

Codebeispiele

Weitere Funktionen, direkt im Code

Laden und Speichern von ndarrays
Direkt im Code

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