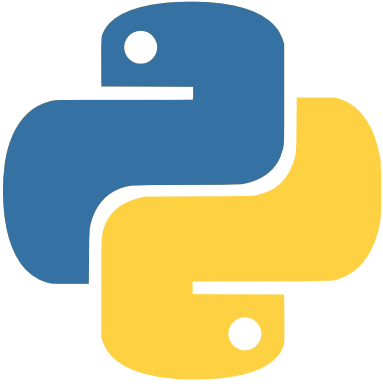
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

matplotlib: Einleitung, Funktionen, Legenden

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

matplotlib: Einleitung, Funktionen, Legenden

Einleitung

- Plotten und Visualisieren von Funktionsverläufen und anderen Daten

Einleitung

- Plotten und Visualisieren von Funktionsverläufen und anderen Daten
- Alltägliche Aufgabe in sehr vielen Disziplinen

Einleitung

- Plotten und Visualisieren von Funktionsverläufen und anderen Daten
- Alltägliche Aufgabe in sehr vielen Disziplinen
- Hier **matplotlib**

Einleitung

- Plotten und Visualisieren von Funktionsverläufen und anderen Daten
- Alltägliche Aufgabe in sehr vielen Disziplinen
- Hier **matplotlib**

- Bereits in Anaconda enthalten, sonst `pip install matplotlib`

Grundlagen

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`
- Mathematisch: **Polygonzüge**

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`
- Mathematisch: **Polygonzüge**
- Einfachstmögliche Nutzung von `plt.plot()`

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`
- Mathematisch: **Polygonzüge**
- Einfachstmögliche Nutzung von `plt.plot()`
  - Übergabe einer einzelnen Liste oder eines einzelnen NumPy-ndarray

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`
- Mathematisch: **Polygonzüge**
- Einfachstmögliche Nutzung von `plt.plot()`
  - Übergabe einer einzelnen Liste oder eines einzelnen NumPy-ndarray
  - Werte interpretiert als **Ordinaten** („y-Werte“) zu **Abszissen** („x-Werte“) 0,1,2,...

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`
- Mathematisch: **Polygonzüge**
- Einfachstmögliche Nutzung von `plt.plot()`
  - Übergabe einer einzelnen Liste oder eines einzelnen NumPy-ndarray
  - Werte interpretiert als **Ordinaten** („y-Werte“) zu **Abszissen** („x-Werte“) 0,1,2,...
- Wichtig: Plots immer „inkrementell zusammengebaut“

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`
- Mathematisch: **Polygonzüge**
- Einfachstmögliche Nutzung von `plt.plot()`
  - Übergabe einer einzelnen Liste oder eines einzelnen NumPy-ndarray
  - Werte interpretiert als **Ordinaten** („y-Werte“) zu **Abszissen** („x-Werte“) 0,1,2,...
- Wichtig: Plots immer „inkrementell zusammengebaut“
  - Erst `plt.show()` leistet Anzeige

Grundlagen

- Konvention: `import matplotlib.pyplot as plt`
- Plots sind „diskret“, d.h. **Datenpunkte verbunden durch Linien** bei der Kernfunktion `plt.plot()`
- Mathematisch: **Polygonzüge**
- Einfachstmögliche Nutzung von `plt.plot()`
  - Übergabe einer einzelnen Liste oder eines einzelnen NumPy-ndarray
  - Werte interpretiert als **Ordinaten** („y-Werte“) zu **Abszissen** („x-Werte“) 0,1,2,...
- Wichtig: Plots immer „inkrementell zusammengebaut“
  - Erst `plt.show()` leistet Anzeige
- In manchen Python-Umgebungen: „magic commands“ nötig, s. Notebook

Codebeispiele

Plotten von Funktionen

Plotten von Funktionen

- Berechnung von Datenpunkten, Interpretation als Paare von Abszissen und Ordinaten, Visualisierung durch Polygonzug

Plotten von Funktionen

- Berechnung von Datenpunkten, Interpretation als Paare von Abszissen und Ordinaten, Visualisierung durch Polygonzug
- `plt.plot()` akzeptiert zwei Listen oder NumPy-ndarrays

Plotten von Funktionen

- Berechnung von Datenpunkten, Interpretation als Paare von Abszissen und Ordinaten, Visualisierung durch Polygonzug
- `plt.plot()` akzeptiert zwei Listen oder NumPy-ndarrays
- Enthalten jeweils gleich viele Einträge

Plotten von Funktionen

- Berechnung von Datenpunkten, Interpretation als Paare von Abszissen und Ordinaten, Visualisierung durch Polygonzug
- `plt.plot()` akzeptiert zwei Listen oder NumPy-ndarrays
- Enthalten jeweils gleich viele Einträge
- **Erste Liste: Abszissen, zweite Liste Ordinaten**

Plotten von Funktionen

- Berechnung von Datenpunkten, Interpretation als Paare von Abszissen und Ordinaten, Visualisierung durch Polygonzug
- `plt.plot()` akzeptiert zwei Listen oder NumPy-ndarrays
- Enthalten jeweils gleich viele Einträge
- **Erste Liste: Abszissen, zweite Liste Ordinaten**
- Paarung über gemeinsamen Index

Plotten von Funktionen

- Berechnung von Datenpunkten, Interpretation als Paare von Abszissen und Ordinaten, Visualisierung durch Polygonzug
- `plt.plot()` akzeptiert zwei Listen oder NumPy-ndarrays
- Enthalten jeweils gleich viele Einträge
- **Erste Liste: Abszissen, zweite Liste Ordinaten**
- Paarung über gemeinsamen Index
- Abszissen: oft mit `numpy.linspace()`

Plotten von Funktionen

- Berechnung von Datenpunkten, Interpretation als Paare von Abszissen und Ordinaten, Visualisierung durch Polygonzug
- `plt.plot()` akzeptiert zwei Listen oder NumPy-ndarrays
- Enthalten jeweils gleich viele Einträge
- **Erste Liste: Abszissen, zweite Liste Ordinaten**
- Paarung über gemeinsamen Index
- Abszissen: oft mit `numpy.linspace()`
- Ordinaten: oft mit vektorisierten NumPy-Funktionen

Codebeispiele

Schönere Plots durch Beschriftungen und Legenden

Schönere Plots durch Beschriftungen und Legenden

- Funktionen: `plt.title()`, `plt.ylabel()`, `plt.xlabel()`

Schönere Plots durch Beschriftungen und Legenden

- Funktionen: `plt.title()`, `plt.ylabel()`, `plt.xlabel()`
- Label als drittes Argument von `plt.plot()`

Schönere Plots durch Beschriftungen und Legenden

- Funktionen: `plt.title()`, `plt.ylabel()`, `plt.xlabel()`
- Label als drittes Argument von `plt.plot()`
- Argumente jeweils normale Zeichenketten

Schönere Plots durch Beschriftungen und Legenden

- Funktionen: `plt.title()`, `plt.ylabel()`, `plt.xlabel()`
- Label als drittes Argument von `plt.plot()`
- Argumente jeweils normale Zeichenketten
- `plt.legend()` schaltet Legende explizit an

Schönere Plots durch Beschriftungen und Legenden

- Funktionen: `plt.title()`, `plt.ylabel()`, `plt.xlabel()`
- Label als drittes Argument von `plt.plot()`
- Argumente jeweils normale Zeichenketten
- `plt.legend()` schaltet Legende explizit an
- Optionales Argument `loc` zur Positionierung

Schönere Plots durch Beschriftungen und Legenden

- Funktionen: `plt.title()`, `plt.ylabel()`, `plt.xlabel()`
- Label als drittes Argument von `plt.plot()`
- Argumente jeweils normale Zeichenketten
- `plt.legend()` schaltet Legende explizit an
- Optionales Argument `loc` zur Positionierung
  - `'best', 'upper right' , 'upper left', 'lower left', 'lower right', 'right', 'center left', 'center right', 'lower center', 'upper center', 'center'`

Codebeispiel

Anpassung des dargestellten Bereichs

Anpassung des dargestellten Bereichs

- `plt.xlim()` und `plt.ylim()`

Anpassung des dargestellten Bereichs

- `plt.xlim()` und `plt.ylim()`
- **Default**: maximaler y-Bereich als Maximum der darzustellenden Werte über dem gegebenen x-Bereich

Anpassung des dargestellten Bereichs

- `plt.xlim()` und `plt.ylim()`
- **Default**: maximaler y-Bereich als Maximum der darzustellenden Werte über dem gegebenen x-Bereich
- Argumente: **Intervallgrenzen**, Reihenfolge „links-rechts“ bzw. „unten-oben“

Anpassung des dargestellten Bereichs

- `plt.xlim()` und `plt.ylim()`
- **Default**: maximaler y-Bereich als Maximum der darzustellenden Werte über dem gegebenen x-Bereich
- Argumente: **Intervallgrenzen**, Reihenfolge „links-rechts“ bzw. „unten-oben“
- Alternativ: Tupel-Argument

Anpassung des dargestellten Bereichs

- `plt.xlim()` und `plt.ylim()`
- **Default**: maximaler y-Bereich als Maximum der darzustellenden Werte über dem gegebenen x-Bereich
- Argumente: **Intervallgrenzen**, Reihenfolge „links-rechts“ bzw. „unten-oben“
- Alternativ: Tupel-Argument
- Funktionen müssen **zwingend beide genutzt** werden

Anpassung des dargestellten Bereichs

- `plt.xlim()` und `plt.ylim()`
- **Default**: maximaler y-Bereich als Maximum der darzustellenden Werte über dem gegebenen x-Bereich
- Argumente: **Intervallgrenzen**, Reihenfolge „links-rechts“ bzw. „unten-oben“
- Alternativ: Tupel-Argument
- Funktionen müssen **zwingend beide genutzt** werden
- Aufruf ohne Argumente: Rückgabe des aktuell verwendete Wertebereichs

Codebeispiel

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