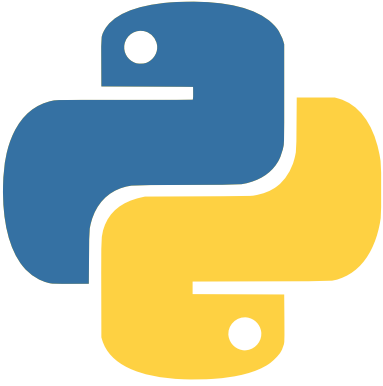
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Pakete und Paketverwaltung

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

CC BY

Pakete und Paketverwaltung

Pakete und Paketverwaltung

- Bereits bekannt: **Standardbibliothek**, große Anzahl an Modulen für vielfältigste Anwendungsbereiche

Pakete und Paketverwaltung

- Bereits bekannt: **Standardbibliothek**, große Anzahl an Modulen für vielfältigste Anwendungsbereiche
  - Dateiverarbeitung: `csv, ...`

Pakete und Paketverwaltung

- Bereits bekannt: **Standardbibliothek**, große Anzahl an Modulen für vielfältigste Anwendungsbereiche
  - Dateiverarbeitung: `csv, ...`
  - Einfache Mathematikfunktionen: `math, cmath, fractions, statistics, random`

Pakete und Paketverwaltung

- Bereits bekannt: **Standardbibliothek**, große Anzahl an Modulen für vielfältigste Anwendungsbereiche
  - Dateiverarbeitung: `csv, ...`
  - Einfache Mathematikfunktionen: `math, cmath, fractions, statistics, random`
  - Datum / Uhrzeit / Zeitmessung: `time`

Pakete und Paketverwaltung

- Bereits bekannt: **Standardbibliothek**, große Anzahl an Modulen für vielfältigste Anwendungsbereiche
  - Dateiverarbeitung: `csv, ...`
  - Einfache Mathematikfunktionen: `math, cmath, fractions, statistics, random`
  - Datum / Uhrzeit / Zeitmessung: `time`
  - Betriebssystem-Zugriff: `os`, `sys`

Pakete und Paketverwaltung

- Bereits bekannt: **Standardbibliothek**, große Anzahl an Modulen für vielfältigste Anwendungsbereiche
  - Dateiverarbeitung: `csv, ...`
  - Einfache Mathematikfunktionen: `math, cmath, fractions, statistics, random`
  - Datum / Uhrzeit / Zeitmessung: `time`
  - Betriebssystem-Zugriff: `os`, `sys`
- In diesem Abschnitt: **Pakete, und Unterscheidung zu Modulen**

Was sind Pakete?

Was sind Pakete?

- Konzeptionelle **Möglichkeit zur Erweiterung** nicht des Sprachumfangs, sondern der Funktionalität bei Nutzung von Python für spezielle Aufgaben

Was sind Pakete?

- Konzeptionelle **Möglichkeit zur Erweiterung** nicht des Sprachumfangs, sondern der Funktionalität bei Nutzung von Python für spezielle Aufgaben
- Technisch zunächst nichts anderes als eine **Zusammenfassung** mehrerer Module unter einer einheitlichen Schnittstelle

Was sind Pakete?

- Konzeptionelle **Möglichkeit zur Erweiterung** nicht des Sprachumfangs, sondern der Funktionalität bei Nutzung von Python für spezielle Aufgaben
- Technisch zunächst nichts anderes als eine **Zusammenfassung** mehrerer Module unter einer einheitlichen Schnittstelle
  - Paket `scipy`: Module für Fragestellungen des Wissenschaftlichen Rechnens

Was sind Pakete?

- Konzeptionelle **Möglichkeit zur Erweiterung** nicht des Sprachumfangs, sondern der Funktionalität bei Nutzung von Python für spezielle Aufgaben
- Technisch zunächst nichts anderes als eine **Zusammenfassung** mehrerer Module unter einer einheitlichen Schnittstelle
  - Paket `scipy`: Module für Fragestellungen des Wissenschaftlichen Rechnens
  - Unterpaket `optimize`: Module für Fragestellungen der kontinuierlichen Optimierung

Was sind Pakete?

- Konzeptionelle **Möglichkeit zur Erweiterung** nicht des Sprachumfangs, sondern der Funktionalität bei Nutzung von Python für spezielle Aufgaben
- Technisch zunächst nichts anderes als eine **Zusammenfassung** mehrerer Module unter einer einheitlichen Schnittstelle
  - Paket `scipy`: Module für Fragestellungen des Wissenschaftlichen Rechnens
  - Unterpaket `optimize`: Module für Fragestellungen der kontinuierlichen Optimierung
  - Darin Funktion `fsolve()`: Nullstellenberechnung

Was sind Pakete?

- Konzeptionelle **Möglichkeit zur Erweiterung** nicht des Sprachumfangs, sondern der Funktionalität bei Nutzung von Python für spezielle Aufgaben
- Technisch zunächst nichts anderes als eine **Zusammenfassung** mehrerer Module unter einer einheitlichen Schnittstelle
  - Paket `scipy`: Module für Fragestellungen des Wissenschaftlichen Rechnens
  - Unterpaket `optimize`: Module für Fragestellungen der kontinuierlichen Optimierung
  - Darin Funktion `fsolve()`: Nullstellenberechnung
  - Paket `numpy`, Unterpaket `linalg`: Module für Matrixrechnung

Was sind Pakete?

- Konzeptionelle **Möglichkeit zur Erweiterung** nicht des Sprachumfangs, sondern der Funktionalität bei Nutzung von Python für spezielle Aufgaben
- Technisch zunächst nichts anderes als eine **Zusammenfassung** mehrerer Module unter einer einheitlichen Schnittstelle
  - Paket `scipy`: Module für Fragestellungen des Wissenschaftlichen Rechnens
  - Unterpaket `optimize`: Module für Fragestellungen der kontinuierlichen Optimierung
  - Darin Funktion `fsolve()`: Nullstellenberechnung
  - Paket `numpy`, Unterpaket `linalg`: Module für Matrixrechnung
  - Darin Funktion `solve()`: Lösung des linearen Gleichungssystems $Ax = b$

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation
  - Veröffentlichung von Programme (OpenSource!)

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation
  - Veröffentlichung von Programme (OpenSource!)
- Mit **Kenntnisstand dieses Crashkurses**: seltenst bis nie eigene Pakete, im Gegensatz zu Modulen

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation
  - Veröffentlichung von Programme (OpenSource!)
- Mit **Kenntnisstand dieses Crashkurses**: seltenst bis nie eigene Pakete, im Gegensatz zu Modulen
- Trotzdem hilfreich: **Verinnerlichung des Grundkonzepts** von Paketen

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation
  - Veröffentlichung von Programme (OpenSource!)
- Mit **Kenntnisstand dieses Crashkurses**: seltenst bis nie eigene Pakete, im Gegensatz zu Modulen
- Trotzdem hilfreich: **Verinnerlichung des Grundkonzepts** von Paketen
  - Sehr aktive Python-Community

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation
  - Veröffentlichung von Programme (OpenSource!)
- Mit **Kenntnisstand dieses Crashkurses**: seltenst bis nie eigene Pakete, im Gegensatz zu Modulen
- Trotzdem hilfreich: **Verinnerlichung des Grundkonzepts** von Paketen
  - Sehr aktive Python-Community
  - Unzählige Pakete für nützliche und weniger nützliche Anwendungszwecke

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation
  - Veröffentlichung von Programme (OpenSource!)
- Mit **Kenntnisstand dieses Crashkurses**: seltenst bis nie eigene Pakete, im Gegensatz zu Modulen
- Trotzdem hilfreich: **Verinnerlichung des Grundkonzepts** von Paketen
  - Sehr aktive Python-Community
  - Unzählige Pakete für nützliche und weniger nützliche Anwendungszwecke
  - Große Arbeitsersparnis für uns als Programmierer*innen

Anwendungsmöglichkeiten für Pakete

- **Anwendungsmöglichkeiten** für Pakete
  - Sehr umfangreiche Programme
  - Logisch zusammenhängende Komponenten zur besseren Code-Organisation
  - Veröffentlichung von Programme (OpenSource!)
- Mit **Kenntnisstand dieses Crashkurses**: seltenst bis nie eigene Pakete, im Gegensatz zu Modulen
- Trotzdem hilfreich: **Verinnerlichung des Grundkonzepts** von Paketen
  - Sehr aktive Python-Community
  - Unzählige Pakete für nützliche und weniger nützliche Anwendungszwecke
  - Große Arbeitsersparnis für uns als Programmierer*innen
  - Nicht ständig das Rad neu erfinden (neu programmieren)

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, . . .

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, . . .
- `matplotlib`: visuell ansprechende hochwertige Plots

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, ...
- `matplotlib`: visuell ansprechende hochwertige Plots
- `SymPy`: symbolisches Rechnen und Computeralgebra, d.h. Lösung von Gleichungen, symbolische Differential- und Integralrechnung, ...

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, ...
- `matplotlib`: visuell ansprechende hochwertige Plots
- `SymPy`: symbolisches Rechnen und Computeralgebra, d.h. Lösung von Gleichungen, symbolische Differential- und Integralrechnung, ...
- `pandas`: Verarbeitung von Daten beliebiger Komplexität, Import/Export für verschiende Formate

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, ...
- `matplotlib`: visuell ansprechende hochwertige Plots
- `SymPy`: symbolisches Rechnen und Computeralgebra, d.h. Lösung von Gleichungen, symbolische Differential- und Integralrechnung, ...
- `pandas`: Verarbeitung von Daten beliebiger Komplexität, Import/Export für verschiende Formate
- `SciPy`: numerische Methoden der Analysis: Differentialgleichungen, Optimierungsprobleme, ...

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, ...
- `matplotlib`: visuell ansprechende hochwertige Plots
- `SymPy`: symbolisches Rechnen und Computeralgebra, d.h. Lösung von Gleichungen, symbolische Differential- und Integralrechnung, ...
- `pandas`: Verarbeitung von Daten beliebiger Komplexität, Import/Export für verschiende Formate
- `SciPy`: numerische Methoden der Analysis: Differentialgleichungen, Optimierungsprobleme, ...
- `SAGE`: fortgeschrittene Computeralgebra und Gruppentheorie

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, ...
- `matplotlib`: visuell ansprechende hochwertige Plots
- `SymPy`: symbolisches Rechnen und Computeralgebra, d.h. Lösung von Gleichungen, symbolische Differential- und Integralrechnung, ...
- `pandas`: Verarbeitung von Daten beliebiger Komplexität, Import/Export für verschiende Formate
- `SciPy`: numerische Methoden der Analysis: Differentialgleichungen, Optimierungsprobleme, ...
- `SAGE`: fortgeschrittene Computeralgebra und Gruppentheorie
- `pyTorch` und `TensorFlow`: Machine Learning, insbesondere Deep Neural Networks

Beispiele für Pakete

- `JupyterLab`: Jupyter Notebooks
- `NumPy`: numerische lineare Algebra, d.h. Normen, Skalarprodukte, lineare Gleichungssysteme, Eigenwertprobleme, ...
- `matplotlib`: visuell ansprechende hochwertige Plots
- `SymPy`: symbolisches Rechnen und Computeralgebra, d.h. Lösung von Gleichungen, symbolische Differential- und Integralrechnung, ...
- `pandas`: Verarbeitung von Daten beliebiger Komplexität, Import/Export für verschiende Formate
- `SciPy`: numerische Methoden der Analysis: Differentialgleichungen, Optimierungsprobleme, ...
- `SAGE`: fortgeschrittene Computeralgebra und Gruppentheorie
- `pyTorch` und `TensorFlow`: Machine Learning, insbesondere Deep Neural Networks
- `pyGame`: graphische Benutzeroberflächen mit Interaktionsmöglichkeiten

Import von Paketen

- Pakete: Sammlungen von Modulen, eventuell „verpackt“ in Unterpakete

Import von Paketen

- Pakete: Sammlungen von Modulen, eventuell „verpackt“ in Unterpakete
- Import einzelner Module und gezielt Funktionen

Import von Paketen

- Pakete: Sammlungen von Modulen, eventuell „verpackt“ in Unterpakete
- Import einzelner Module und gezielt Funktionen
- Ohne das gesamte Paket verstehen zu müssen

Import von Paketen

- Pakete: Sammlungen von Modulen, eventuell „verpackt“ in Unterpakete
- Import einzelner Module und gezielt Funktionen
- Ohne das gesamte Paket verstehen zu müssen
- `import scipy.optimize; help(scipy.optimize.fsolve)`

Import von Paketen

- Pakete: Sammlungen von Modulen, eventuell „verpackt“ in Unterpakete
- Import einzelner Module und gezielt Funktionen
- Ohne das gesamte Paket verstehen zu müssen
- `import scipy.optimize; help(scipy.optimize.fsolve)`
- `from scipy.optimize import fsolve as nullstellenfinder`

Paketverwaltung

Paketverwaltung

- Anaconda: kuratierte Sammlung von Paketen mit Fokus auf Data Science

Paketverwaltung

- Anaconda: kuratierte Sammlung von Paketen mit Fokus auf Data Science
- Vermutlich im Studium vollständig ausreichend

Paketverwaltung

- Anaconda: kuratierte Sammlung von Paketen mit Fokus auf Data Science
- Vermutlich im Studium vollständig ausreichend
- Falls nicht: `conda` als Tool zur Nachinstallation von Paketen

Paketverwaltung

- Anaconda: kuratierte Sammlung von Paketen mit Fokus auf Data Science
- Vermutlich im Studium vollständig ausreichend
- Falls nicht: `conda` als Tool zur Nachinstallation von Paketen
- Greift auf PYPI zu, Python Package Index

Paketverwaltung

- Anaconda: kuratierte Sammlung von Paketen mit Fokus auf Data Science
- Vermutlich im Studium vollständig ausreichend
- Falls nicht: `conda` als Tool zur Nachinstallation von Paketen
- Greift auf PYPI zu, Python Package Index
- 448.825 Pakete, Stand 22. April 2023

Paketverwaltung

- Anaconda: kuratierte Sammlung von Paketen mit Fokus auf Data Science
- Vermutlich im Studium vollständig ausreichend
- Falls nicht: `conda` als Tool zur Nachinstallation von Paketen
- Greift auf PYPI zu, Python Package Index
- 448.825 Pakete, Stand 22. April 2023
- Hinweise zu anderen Python-Installationen im Jupyter Notebook

Erstellung eigener Pakete

Erstellung eigener Pakete

- Fortgeschrittenes Thema

Erstellung eigener Pakete

- Fortgeschrittenes Thema
- Bei Interesse: Details und Beispiel im Jupyter Notebook

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

CC BY

Weitere Quellen:

- Logos Universität Stuttgart, IANS, SimTech: Universität Stuttgart, alle Rechte vorbehalten
- Logo Python: https://freesvg.org/387, CC-0
- Logo Stiftung: Stiftung Innovation in der Hochschullehre, alle Rechte vorbehalten
- Logo ZOERR: Universität Tübingen, alle Rechte vorbehalten

ZOERR

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