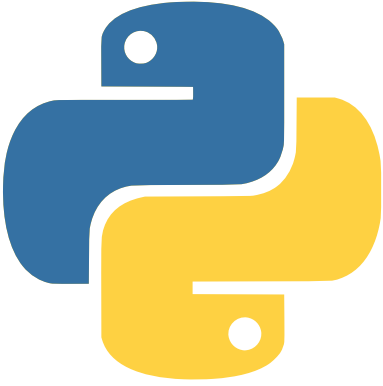
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Einführung in Datentypen

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

Einführung in Datentypen

Einführung in Datentypen

- Variablen haben einen festen sogenannten **Datentyp**

Einführung in Datentypen

- Variablen haben einen festen sogenannten **Datentyp**
- Macht Sinn: nicht alle Datentypen sind kompatibel mit allen Operationen

Einführung in Datentypen

- Variablen haben einen festen sogenannten **Datentyp**
- Macht Sinn: nicht alle Datentypen sind kompatibel mit allen Operationen
  - Beispiel: Addition der Zahl 42 auf 10 TB KI-Lerndaten ???

Einführung in Datentypen

- Variablen haben einen festen sogenannten **Datentyp**
- Macht Sinn: nicht alle Datentypen sind kompatibel mit allen Operationen
  - Beispiel: Addition der Zahl 42 auf 10 TB KI-Lerndaten ???
- Python ist **dynamisch typisiert**

Einführung in Datentypen

- Variablen haben einen festen sogenannten **Datentyp**
- Macht Sinn: nicht alle Datentypen sind kompatibel mit allen Operationen
  - Beispiel: Addition der Zahl 42 auf 10 TB KI-Lerndaten ???
- Python ist **dynamisch typisiert**
  - Datentyp einer Variable wird durch Datentyp des Ausdrucks auf der rechten Seite einer Zuweisung festgelegt

Einführung in Datentypen

- Variablen haben einen festen sogenannten **Datentyp**
- Macht Sinn: nicht alle Datentypen sind kompatibel mit allen Operationen
  - Beispiel: Addition der Zahl 42 auf 10 TB KI-Lerndaten ???
- Python ist **dynamisch typisiert**
  - Datentyp einer Variable wird durch Datentyp des Ausdrucks auf der rechten Seite einer Zuweisung festgelegt
- Befehl `type(variable)` liefert den Typ einer Variable

Wir experimentieren ein wenig . . .

Zurück zur Einführung in Datentypen

Eine erste Einführung in Datentypen

- Andere Programmiersprachen sind statisch typisiert

Eine erste Einführung in Datentypen

- Andere Programmiersprachen sind statisch typisiert
  - **Pro:** dynamische Typisierung kann Nachdenken ersparen

Eine erste Einführung in Datentypen

- Andere Programmiersprachen sind statisch typisiert
  - **Pro:** dynamische Typisierung kann Nachdenken ersparen
  - **Con:** dynamische Typisierung fördert u.U. schlechten Programmierstil, Wiederverwendung von Variablen unterschieblichen Typs

Eine erste Einführung in Datentypen

- Andere Programmiersprachen sind statisch typisiert
  - **Pro:** dynamische Typisierung kann Nachdenken ersparen
  - **Con:** dynamische Typisierung fördert u.U. schlechten Programmierstil, Wiederverwendung von Variablen unterschieblichen Typs
  - Keine Wertung, stattdessen Aufbau von Verständnis der dynamischen Typisierung

Eine erste Einführung in Datentypen

- Andere Programmiersprachen sind statisch typisiert
  - **Pro:** dynamische Typisierung kann Nachdenken ersparen
  - **Con:** dynamische Typisierung fördert u.U. schlechten Programmierstil, Wiederverwendung von Variablen unterschieblichen Typs
  - Keine Wertung, stattdessen Aufbau von Verständnis der dynamischen Typisierung
- **Quiz:** Überprüfen Sie das folgende Beispiel auf die verwendeten Typen

```
a = 5 + 4; a = "Luke, ich bin" + "Dein Vater"
```

Eine erste Einführung in Datentypen

- Andere Programmiersprachen sind statisch typisiert
  - **Pro:** dynamische Typisierung kann Nachdenken ersparen
  - **Con:** dynamische Typisierung fördert u.U. schlechten Programmierstil, Wiederverwendung von Variablen unterschieblichen Typs
  - Keine Wertung, stattdessen Aufbau von Verständnis der dynamischen Typisierung
- **Quiz:** Überprüfen Sie das folgende Beispiel auf die verwendeten Typen

```
a = 5 + 4; a = "Luke, ich bin" + "Dein Vater"
```

  - Überlegen Sie sich den Datentyp der Eingaben und der Ergebnisse

Eine erste Einführung in Datentypen

- Andere Programmiersprachen sind statisch typisiert
  - **Pro:** dynamische Typisierung kann Nachdenken ersparen
  - **Con:** dynamische Typisierung fördert u.U. schlechten Programmierstil, Wiederverwendung von Variablen unterschieblichen Typs
  - Keine Wertung, stattdessen Aufbau von Verständnis der dynamischen Typisierung
- **Quiz:** Überprüfen Sie das folgende Beispiel auf die verwendeten Typen

```
a = 5 + 4; a = "Luke, ich bin" + "Dein Vater"
```

  - Überlegen Sie sich den Datentyp der Eingaben und der Ergebnisse
  - Validieren Sie Ihre These mit `type()`

Grundlegende (builtin) Datentypen und Typ-Konvertierung

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`
  - Numerische Datentypen: `int`, `float`, `complex`

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`
  - Numerische Datentypen: `int`, `float`, `complex`
  - Sequenzen: `list`

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`
  - Numerische Datentypen: `int`, `float`, `complex`
  - Sequenzen: `list`
  - Zeichenketten: `str`

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`
  - Numerische Datentypen: `int`, `float`, `complex`
  - Sequenzen: `list`
  - Zeichenketten: `str`
  - n-Tupel: `tuple`

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`
  - Numerische Datentypen: `int`, `float`, `complex`
  - Sequenzen: `list`
  - Zeichenketten: `str`
  - n-Tupel: `tuple`
  - Und viele mehr, Details in späteren Lerneinheiten

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`
  - Numerische Datentypen: `int`, `float`, `complex`
  - Sequenzen: `list`
  - Zeichenketten: `str`
  - n-Tupel: `tuple`
  - Und viele mehr, Details in späteren Lerneinheiten
- Typ der Daten legt fest, wie sich Operationen verhalten

Grundlegende (builtin) Datentypen

- Python bietet **viele eingebaute Datentypen**, bspw.
  - Logische Variablen: `bool`
  - Numerische Datentypen: `int`, `float`, `complex`
  - Sequenzen: `list`
  - Zeichenketten: `str`
  - n-Tupel: `tuple`
  - Und viele mehr, Details in späteren Lerneinheiten
- Typ der Daten legt fest, wie sich Operationen verhalten
- Dazu: **explizite und implizite Typ-Konvertierungen**

Wir experimentieren direkt im Code …

Zusammenfassung: Typ-Konvertierungen

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**
  - Immer der sichere Weg, selbst wenn unnötig

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**
  - Immer der sichere Weg, selbst wenn unnötig
  - Vermeidet unnötige Überraschungen und qualvolle Fehlersuche

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**
  - Immer der sichere Weg, selbst wenn unnötig
  - Vermeidet unnötige Überraschungen und qualvolle Fehlersuche
- **Implizite Konvertierungen**

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**
  - Immer der sichere Weg, selbst wenn unnötig
  - Vermeidet unnötige Überraschungen und qualvolle Fehlersuche
- **Implizite Konvertierungen**
  - Kenntnis spart viel Zeit, „sich drauf verlassen“

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**
  - Immer der sichere Weg, selbst wenn unnötig
  - Vermeidet unnötige Überraschungen und qualvolle Fehlersuche
- **Implizite Konvertierungen**
  - Kenntnis spart viel Zeit, „sich drauf verlassen“
- **Pragmatische Daumenregel**

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**
  - Immer der sichere Weg, selbst wenn unnötig
  - Vermeidet unnötige Überraschungen und qualvolle Fehlersuche
- **Implizite Konvertierungen**
  - Kenntnis spart viel Zeit, „sich drauf verlassen“
- **Pragmatische Daumenregel**
  - Im Zweifel mit `print/type` Einzeiler ausprobieren, ob es etwas implizites gibt

Zusammenfassung: Typ-Konvertierungen

- **Explizite Konvertierungen**
  - Immer der sichere Weg, selbst wenn unnötig
  - Vermeidet unnötige Überraschungen und qualvolle Fehlersuche
- **Implizite Konvertierungen**
  - Kenntnis spart viel Zeit, „sich drauf verlassen“
- **Pragmatische Daumenregel**
  - Im Zweifel mit `print/type` Einzeiler ausprobieren, ob es etwas implizites gibt
  - Bei verbleibenden Zweifeln: besser explizit konvertieren

Beispiel: Strings
Ein- und Ausgabe

Beispiel: Strings, Ein- und Ausgabe

- Datentyp `str` (für string) speichert **Zeichenketten**

Beispiel: Strings, Ein- und Ausgabe

- Datentyp `str` (für string) speichert **Zeichenketten**
- Initialisierung: `s = "mein erster string"`

Beispiel: Strings, Ein- und Ausgabe

- Datentyp `str` (für string) speichert **Zeichenketten**
- Initialisierung: `s = "mein erster string"`
- **Konkatenation von Strings** mit dem Plus-Operator

Beispiel: Strings, Ein- und Ausgabe

- Datentyp `str` (für string) speichert **Zeichenketten**
- Initialisierung: `s = "mein erster string"`
- **Konkatenation von Strings** mit dem Plus-Operator
- **Länge eines Strings** mit dem Befehl `len()`

Beispiel: Strings, Ein- und Ausgabe

- Datentyp `str` (für string) speichert **Zeichenketten**
- Initialisierung: `s = "mein erster string"`
- **Konkatenation von Strings** mit dem Plus-Operator
- **Länge eines Strings** mit dem Befehl `len()`
- **Mehrfaches Hintereinanderhängen** mit `*n`, `n` ist gewünschte Anzahl

Wir probieren das aus . . .

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