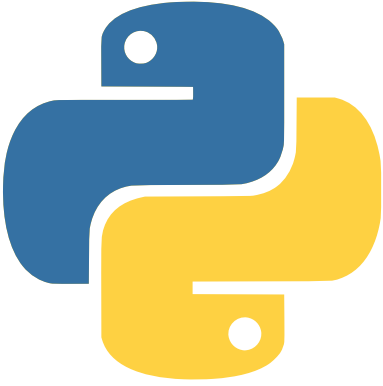
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Der Datentyp int für ganzzahlige Variablen

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

Der Datentyp int
für ganzzahlige Variablen

Der Datentyp int für ganzzahlige Variablen

- Prinzipiell für beliebig große ganze Zahlen verwendbar

Der Datentyp int für ganzzahlige Variablen

- Prinzipiell für beliebig große ganze Zahlen verwendbar
- Nur der verfügbare Arbeitsspeicher des Rechners bildet eine Grenze

Der Datentyp int für ganzzahlige Variablen

- Prinzipiell für beliebig große ganze Zahlen verwendbar
- Nur der verfügbare Arbeitsspeicher des Rechners bildet eine Grenze
- Anders als bei den meisten anderen Programmiersprachen

Der Datentyp int für ganzzahlige Variablen

- Prinzipiell für beliebig große ganze Zahlen verwendbar
- Nur der verfügbare Arbeitsspeicher des Rechners bildet eine Grenze
- Anders als bei den meisten anderen Programmiersprachen
- Mathematisch: Mengen $\mathbb{N}$ und $\mathbb{Z}$ der natürlichen bzw. ganzen Zahlen

Ein kurzes Codebeispiel

Rechenoperationen mit int Variablen

Rechenoperationen mit int Variablen

- Addition: +

Rechenoperationen mit int Variablen

- Addition: `+`
- Subtraktion: `-`

Rechenoperationen mit int Variablen

- Addition: `+`
- Subtraktion: `-`
- Produkt: `*`

Rechenoperationen mit int Variablen

- Addition: `+`
- Subtraktion: `-`
- Produkt: `*`
- Potenz: `**`

Rechenoperationen mit int Variablen

- Addition: `+`
- Subtraktion: `-`
- Produkt: `*`
- Potenz: `**`
- Division: `/`

Rechenoperationen mit int Variablen

- Addition: `+`
- Subtraktion: `-`
- Produkt: `*`
- Potenz: `**`
- Division: `/`
- Ganzzahlige Division: `//`, Division und abrunden auf die nächstkleinere ganze Zahl

Rechenoperationen mit int Variablen

- Addition: `+`
- Subtraktion: `-`
- Produkt: `*`
- Potenz: `**`
- Division: `/`
- Ganzzahlige Division: `//`, Division und abrunden auf die nächstkleinere ganze Zahl
- Modulo: `%`, Rest bei einer ganzzahligen Division

Rechenoperationen mit int Variablen

- Addition: `+`
- Subtraktion: `-`
- Produkt: `*`
- Potenz: `**`
- Division: `/`
- Ganzzahlige Division: `//`, Division und abrunden auf die nächstkleinere ganze Zahl
- Modulo: `%`, Rest bei einer ganzzahligen Division
- **Beispiele**

Vergleichsoperationen mit int Variablen

Vergleichsoperationen mit int Variablen

- echt kleiner: <

Vergleichsoperationen mit int Variablen

- echt kleiner: <
- echt größer: >

Vergleichsoperationen mit int Variablen

- echt kleiner: <
- echt größer: >
- kleiner gleich: <=

Vergleichsoperationen mit int Variablen

- echt kleiner: `<`
- echt größer: `>`
- kleiner gleich: `<=`
- größer gleich: `>=`

Vergleichsoperationen mit int Variablen

- echt kleiner: <
- echt größer: >
- kleiner gleich: <=
- größer gleich: >=
- gleich: ==

Vergleichsoperationen mit int Variablen

- echt kleiner: `<`
- echt größer: `>`
- kleiner gleich: `<=`
- größer gleich: `>=`
- gleich: `==`
- ungleich: `!=`

Vergleichsoperationen mit int Variablen

- echt kleiner: `<`
- echt größer: `>`
- kleiner gleich: `<=`
- größer gleich: `>=`
- gleich: `==`
- ungleich: `!=`
- Beispiele nicht wirklich nötig

Kurzschreibweise für elementare Operationen

Kurzschreibweise für elementare Operationen

- Kurzschreibweise für Inkremente (analog für die anderen Rechenoperationen)

Kurzschreibweise für elementare Operationen

- Kurzschreibweise für Inkremente (analog für die anderen Rechenoperationen)
- Beispiele

```
a = 5
a += 5    # Entspricht a = a + 5
a -= 5    # Entspricht a = a - 5
a *= 5    # Entspricht a = a * 5
a **= 5   # Entspricht a = a ** 5
a /= 5    # Entspricht a = a / 5
a //= 5   # Entspricht a = a // 5
```

Kurzschreibweise für elementare Operationen

- Kurzschreibweise für Inkremente (analog für die anderen Rechenoperationen)
- Beispiele

```
a = 5
a += 5    # Entspricht a = a + 5
a -= 5    # Entspricht a = a - 5
a *= 5    # Entspricht a = a * 5
a **= 5   # Entspricht a = a ** 5
a /= 5    # Entspricht a = a / 5
a //= 5   # Entspricht a = a // 5
```

- Tatsächlich sind Vergleichsoperatoren bereits Kurzschreibweisen

`a != b` ist kurz für `not(a==b)`

Datentypen bei der Division

Datentypen bei der Division

- Division **ändert den Datentyp**

Datentypen bei der Division

- Division **ändert den Datentyp**
- Automatische Konvertierung zu `float`

Datentypen bei der Division

- Division **ändert den Datentyp**
- Automatische Konvertierung zu `float`
- Verhalten entspricht mathematischer Intuition, wenn eine Division „nicht glatt aufgeht“

Datentypen bei der Division

- Division **ändert den Datentyp**
- Automatische Konvertierung zu `float`
- Verhalten entspricht mathematischer Intuition, wenn eine Division „nicht glatt aufgeht“
- Aus Konsistenzgründen: Ergebnis immer vom Typ `float`, auch wenn ganze Zahl

Datentypen bei der Division

- Division **ändert den Datentyp**
- Automatische Konvertierung zu `float`
- Verhalten entspricht mathematischer Intuition, wenn eine Division „nicht glatt aufgeht“
- Aus Konsistenzgründen: Ergebnis immer vom Typ `float`, auch wenn ganze Zahl
- Alles andere würde mehr verwirren

Eine kurze Demonstration

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