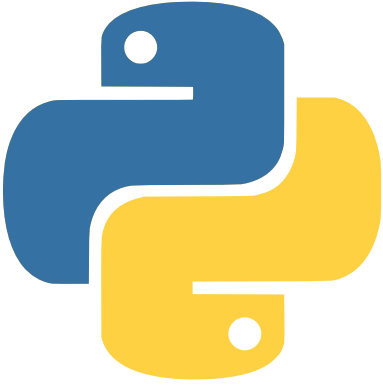
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Kontrollstrukturen: while Schleifen und Blöcke

Dominik Göddeke

Stiftung Innovation in der Hochschullehre

Kontrollstrukturen: while Schleifen und Blöcke

Kontrollstrukturen: Die while Schleife

- In vielen Programmen: **Wiederholung** bestimmter Anweisungen, und/oder **Ausführung nur unter bestimmten Bedingungen**

Kontrollstrukturen: Die while Schleife

- In vielen Programmen: **Wiederholung** bestimmter Anweisungen, und/oder **Ausführung nur unter bestimmten Bedingungen**
- Dafür: **Schleifen** und andere **Kontrollstrukturen**, hier ein erstes Beispiel

Kontrollstrukturen: Die while Schleife

- In vielen Programmen: **Wiederholung** bestimmter Anweisungen, und/oder **Ausführung nur unter bestimmten Bedingungen**
- Dafür: **Schleifen** und andere **Kontrollstrukturen**, hier ein erstes Beispiel
- Motivation: ausgesprochen schlechter Programmierstil

Kontrollstrukturen: Die while Schleife

- In vielen Programmen: **Wiederholung** bestimmter Anweisungen, und/oder **Ausführung nur unter bestimmten Bedingungen**
- Dafür: **Schleifen** und andere **Kontrollstrukturen**, hier ein erstes Beispiel
- Motivation: ausgesprochen schlechter Programmierstil

```
print("Das Quadrat von 1 ist", 1**2)
print("Das Quadrat von 2 ist", 2**2)
print("Das Quadrat von 3 ist", 3**2)
print("Das Quadrat von 4 ist", 4**2)
print("Das Quadrat von 5 ist", 5**2)
print("Das Quadrat von 6 ist", 6**2)
print("Das Quadrat von 7 ist", 7**2)
```

Kontrollstrukturen: Die while Schleife

- **Scharfes Hinsehen**: eine nötige Variable ändert sich auf strukturierte Art

Kontrollstrukturen: Die while Schleife

- **Scharfes Hinsehen**: eine nötige Variable ändert sich auf strukturierte Art

```
print("Das Quadrat von 1 ist", 1**2)
print("Das Quadrat von 2 ist", 2**2)
print("Das Quadrat von 3 ist", 3**2)
print("Das Quadrat von 4 ist", 4**2)
print("Das Quadrat von 5 ist", 5**2)
print("Das Quadrat von 6 ist", 6**2)
print("Das Quadrat von 7 ist", 7**2)
```

Kontrollstrukturen: Die while Schleife

- **Scharfes Hinsehen**: eine nötige Variable ändert sich auf strukturierte Art

```
print("Das Quadrat von 1 ist", 1**2)
print("Das Quadrat von 2 ist", 2**2)
print("Das Quadrat von 3 ist", 3**2)
print("Das Quadrat von 4 ist", 4**2)
print("Das Quadrat von 5 ist", 5**2)
print("Das Quadrat von 6 ist", 6**2)
print("Das Quadrat von 7 ist", 7**2)
```

- Verdacht: `i = i + 1` für eine **Kontrollvariable**

Kontrollstrukturen: Die while Schleife

- **Schärfstes Hinsehen**: Berechnung immer identisch

Kontrollstrukturen: Die while Schleife

- **Schärfstes Hinsehen**: Berechnung immer identisch

```
print("Das Quadrat von 1 ist", 1**2)
print("Das Quadrat von 2 ist", 2**2)
print("Das Quadrat von 3 ist", 3**2)
print("Das Quadrat von 4 ist", 4**2)
print("Das Quadrat von 5 ist", 5**2)
print("Das Quadrat von 6 ist", 6**2)
print("Das Quadrat von 7 ist", 7**2)
```

Kontrollstrukturen: Die while Schleife

- **Schärfstes Hinsehen**: Berechnung immer identisch

```
print("Das Quadrat von 1 ist", 1**2)
print("Das Quadrat von 2 ist", 2**2)
print("Das Quadrat von 3 ist", 3**2)
print("Das Quadrat von 4 ist", 4**2)
print("Das Quadrat von 5 ist", 5**2)
print("Das Quadrat von 6 ist", 6**2)
print("Das Quadrat von 7 ist", 7**2)
```

- Verdacht: `j = i**2` als Verarbeitung der Kontrollvariable

Wir realisieren wir das in Code?

Diskussion des Beispiels

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

- Schleife läuft, bis **Bedingung** $i \leq 7$ erfüllt, d.h. 7 Iterationen

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

- Schleife läuft, bis **Bedingung** $i \leq 7$ erfüllt, d.h. 7 Iterationen
- Essentiell: **Inkrement** $i = i + 1$ innerhalb, sonst **Endlosschleife**

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

- Schleife läuft, bis **Bedingung** $i \leq 7$ erfüllt, d.h. 7 Iterationen
- Essentiell: **Inkrement** $i = i + 1$ innerhalb, sonst **Endlosschleife**
- **Doppelpunkt** startet sogenannten **Block**

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

- Schleife läuft, bis **Bedingung** $i \leq 7$ erfüllt, d.h. 7 Iterationen
- Essentiell: **Inkrement** $i = i + 1$ innerhalb, sonst **Endlosschleife**
- **Doppelpunkt** startet sogenannten **Block**
  - Definiert durch **Einrückung des Quelltexts** mit Leerzeichen

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

- Schleife läuft, bis **Bedingung** $i \leq 7$ erfüllt, d.h. 7 Iterationen
- Essentiell: **Inkrement** $i = i + 1$ innerhalb, sonst **Endlosschleife**
- **Doppelpunkt** startet sogenannten **Block**
  - Definiert durch **Einrückung des Quelltexts** mit Leerzeichen
  - Block endet mit Ende der Einrückung

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

- Schleife läuft, bis **Bedingung** $i \leq 7$ erfüllt, d.h. 7 Iterationen
- Essentiell: **Inkrement** $i = i + 1$ innerhalb, sonst **Endlosschleife**
- **Doppelpunkt** startet sogenannten **Block**
  - Definiert durch **Einrückung des Quelltexts** mit Leerzeichen
  - Block endet mit Ende der Einrückung
  - Codezeile `i=42` nicht Teil des Blocks

Diskussion des Beispiels

```
i = 1
while i <= 7:
  print("Das Quadrat von {:d} ist {:d}".format(i, i**2))
  i = i + 1
i = 42
```

- Schleife läuft, bis **Bedingung** $i \leq 7$ erfüllt, d.h. 7 Iterationen
- Essentiell: **Inkrement** $i = i + 1$ innerhalb, sonst **Endlosschleife**
- **Doppelpunkt** startet sogenannten **Block**
  - Definiert durch **Einrückung des Quelltexts** mit Leerzeichen
  - Block endet mit Ende der Einrückung
  - Codezeile `i=42` nicht Teil des Blocks
- Definition `while` Schleife: Ausführung aller Anweisungen im zugehörigen Block sequentiell, einmal pro Schleifendurchlauf

Blöcke in Python

Blöcke in Python

- Kompletter Block muss mit **identischer Anzahl** Leerzeichen eingerückt werden

Blöcke in Python

- Kompletter Block muss mit **identischer Anzahl** Leerzeichen eingerückt werden
- Konvention: vier Leerzeichen, gute Lesbarkeit

Blöcke in Python

- Kompletter Block muss mit **identischer Anzahl** Leerzeichen eingerückt werden
- Konvention: vier Leerzeichen, gute Lesbarkeit
- Anzahl aber theoretisch beliebig, solange identisch in einem Block

Blöcke in Python

- Kompletter Block muss mit **identischer Anzahl** Leerzeichen eingerückt werden
- Konvention: vier Leerzeichen, gute Lesbarkeit
- Anzahl aber theoretisch beliebig, solange identisch in einem Block
- Leere Zeilen müssen nicht unbedingt eingerückt werden

Blöcke in Python

- Kompletter Block muss mit **identischer Anzahl** Leerzeichen eingerückt werden
- Konvention: vier Leerzeichen, gute Lesbarkeit
- Anzahl aber theoretisch beliebig, solange identisch in einem Block
- Leere Zeilen müssen nicht unbedingt eingerückt werden
- Hilfreich: IDEs bzw. intelligente Editoren rücken automatisch ein

Quiz: Was schlägt hier fehl?

Flexibilität von while Schleifen

Flexibilität von while Schleifen

- Am vorherigen Beispiel auch gesehen: **Flexibilität von while Schleifen**

Flexibilität von while Schleifen

- Am vorherigen Beispiel auch gesehen: **Flexibilität von while Schleifen**
- Vorwärts- und Rückwärtsdurchläufe

Flexibilität von while Schleifen

- Am vorherigen Beispiel auch gesehen: **Flexibilität von while Schleifen**
- Vorwärts- und Rückwärtsdurchläufe
- Beliebige Inkremente und Dekremente der Kontrollvariable

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