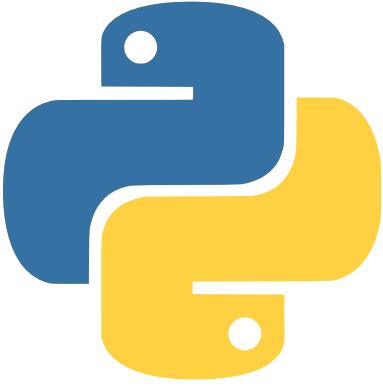
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Variable Anzahl von Funktionsargumenten (positional arguments)

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

CC BY

Variable Anzahl von Funktionsargumenten (positional arguments)

Variable Anzahl von Argumenten

- Oft nützlich: **variable Anzahl an Funktionsargumenten**

Variable Anzahl von Argumenten

- Oft nützlich: **variable Anzahl an Funktionsargumenten**
- Insbesondere wenn initial unklar, wie viele tatsächlich nötig

Variable Anzahl von Argumenten

- Oft nützlich: **variable Anzahl an Funktionsargumenten**
- Insbesondere wenn initial unklar, wie viele tatsächlich nötig
- Dazu: **positional arguments (Positions-Argumente)** für beliebige Anzahl an unbenannten Argumenten

Variable Anzahl von Argumenten

- Oft nützlich: **variable Anzahl an Funktionsargumenten**
- Insbesondere wenn initial unklar, wie viele tatsächlich nötig
- Dazu: **positional arguments (Positions-Argumente)** für beliebige Anzahl an unbenannten Argumenten
- Und: **keyword arguments (Keyword-Argumente)** für beliebige Anzahl benannter Argumente

Positional arguments

Syntax bei positional arguments

- Syntax für Übergabe einer beliebigen Anzahl an Argumente:

```
def function_name(arg1, arg2=5, *args):
```

Syntax bei positional arguments

- Syntax für Übergabe einer beliebigen Anzahl an Argumente:

```
def function_name(arg1, arg2=5, *args):
```

- `arg1`: Platzhalter für eine feste Anzahl an Pflicht-Argumenten

Syntax bei positional arguments

- Syntax für Übergabe einer beliebigen Anzahl an Argumente:

```
def function_name(arg1, arg2=5, *args):
```

- `arg1`: Platzhalter für eine feste Anzahl an Pflicht-Argumenten
- `arg2` analog für (benannte) Argumente mit Standardwerten

Syntax bei positional arguments

- Syntax für Übergabe einer beliebigen Anzahl an Argumente:

```
def function_name(arg1, arg2=5, *args):
```

- `arg1`: Platzhalter für eine feste Anzahl an Pflicht-Argumenten
- `arg2` analog für (benannte) Argumente mit Standardwerten
- `args`: Tupel mit variablen Argumenten

Syntax bei positional arguments

- Syntax für Übergabe einer beliebigen Anzahl an Argumente:

```
def function_name(arg1, arg2=5, *args):
```

- `arg1`: Platzhalter für eine feste Anzahl an Pflicht-Argumenten
- `arg2` analog für (benannte) Argumente mit Standardwerten
- `args`: Tupel mit variablen Argumenten
- Name `args` kann beliebig gewählt werden

Syntax bei positional arguments

- Syntax für Übergabe einer beliebigen Anzahl an Argumente:

```
def function_name(arg1, arg2=5, *args):
```

- `arg1`: Platzhalter für eine feste Anzahl an Pflicht-Argumenten
- `arg2` analog für (benannte) Argumente mit Standardwerten
- `args`: Tupel mit variablen Argumenten
- Name `args` kann beliebig gewählt werden
- Sternchen ist Pflicht

Codebeispiele

Keyword arguments

Syntax bei keyword arguments

```
def function_name(..., **kwargs)
```

- Möglichkeit für benannte Übergabe einer beliebigen Anzahl an Argumenten

Syntax bei keyword arguments

```
def function_name(..., **kwargs)
```

- Möglichkeit für benannte Übergabe einer beliebigen Anzahl an Argumenten
- Pünktchen: beliebige Pflicht- und normale benannte Argumente, und eventuelle positional arguments

Syntax bei keyword arguments

```
def function_name(..., **kwargs)
```

- Möglichkeit für benannte Übergabe einer beliebigen Anzahl an Argumenten
- Pünktchen: beliebige Pflicht- und normale benannte Argumente, und eventuelle positional arguments
- **Merke**: kein Sternchen: Pflicht- und Standardargumente; ein Sternchen: positional arguments; zwei Sternchen: keyword arguments

Syntax bei keyword arguments

```
def function_name(..., **kwargs)
```

- Möglichkeit für benannte Übergabe einer beliebigen Anzahl an Argumenten
- Pünktchen: beliebige Pflicht- und normale benannte Argumente, und eventuelle positional arguments
- **Merke**: kein Sternchen: Pflicht- und Standardargumente; ein Sternchen: positional arguments; zwei Sternchen: keyword arguments
- Sternchen relevant, nicht Name der Keyword-Argumente

Unterschied zu positional arguments

```
def function_name(..., **kwargs)
```

- In `kwargs`: übergebene benannte Variablen in einem Dictionary

Unterschied zu positional arguments

```
def function_name(..., **kwargs)
```

- In `kwargs`: übergebene benannte Variablen in einem Dictionary
- Nicht in ein Tupel wie bei positional arguments

Unterschied zu positional arguments

```
def function_name(..., **kwargs)
```

- In `kwargs`: übergebene benannte Variablen in einem Dictionary
- Nicht in ein Tupel wie bei positional arguments
- Extraktion der Funktionsargumente aus dem Dictionary

Unterschied zu positional arguments

```
def function_name(..., **kwargs)
```

- In `kwargs`: übergebene benannte Variablen in einem Dictionary
- Nicht in ein Tupel wie bei positional arguments
- Extraktion der Funktionsargumente aus dem Dictionary
- Schlüssel: Variablenbenennung aus Aufruf, als String

Unterschied zu positional arguments

```
def function_name(..., **kwargs)
```

- In `kwargs`: übergebene benannte Variablen in einem Dictionary
- Nicht in ein Tupel wie bei positional arguments
- Extraktion der Funktionsargumente aus dem Dictionary
- Schlüssel: Variablenbenennung aus Aufruf, als String
- Zugehörige Werte: Werte aus Aufruf

Codebeispiele

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