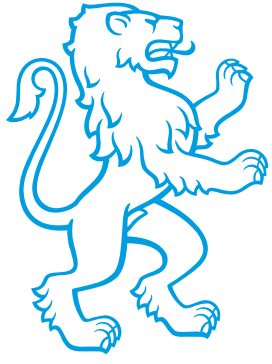


Kantonsschule Im Lee

Informatik: Programmieren

 Kapitel 4: **Funktionen mit `return`**

Woran erinnert Sie dieses Bild?

Funktionen

Womit können Funktionen verglichen werden?

Funktionen

Womit können Funktionen verglichen werden? Weshalb?

Funktionen

Womit können Funktionen verglichen werden? Weshalb?

Funktionierende Funktion? 🐛

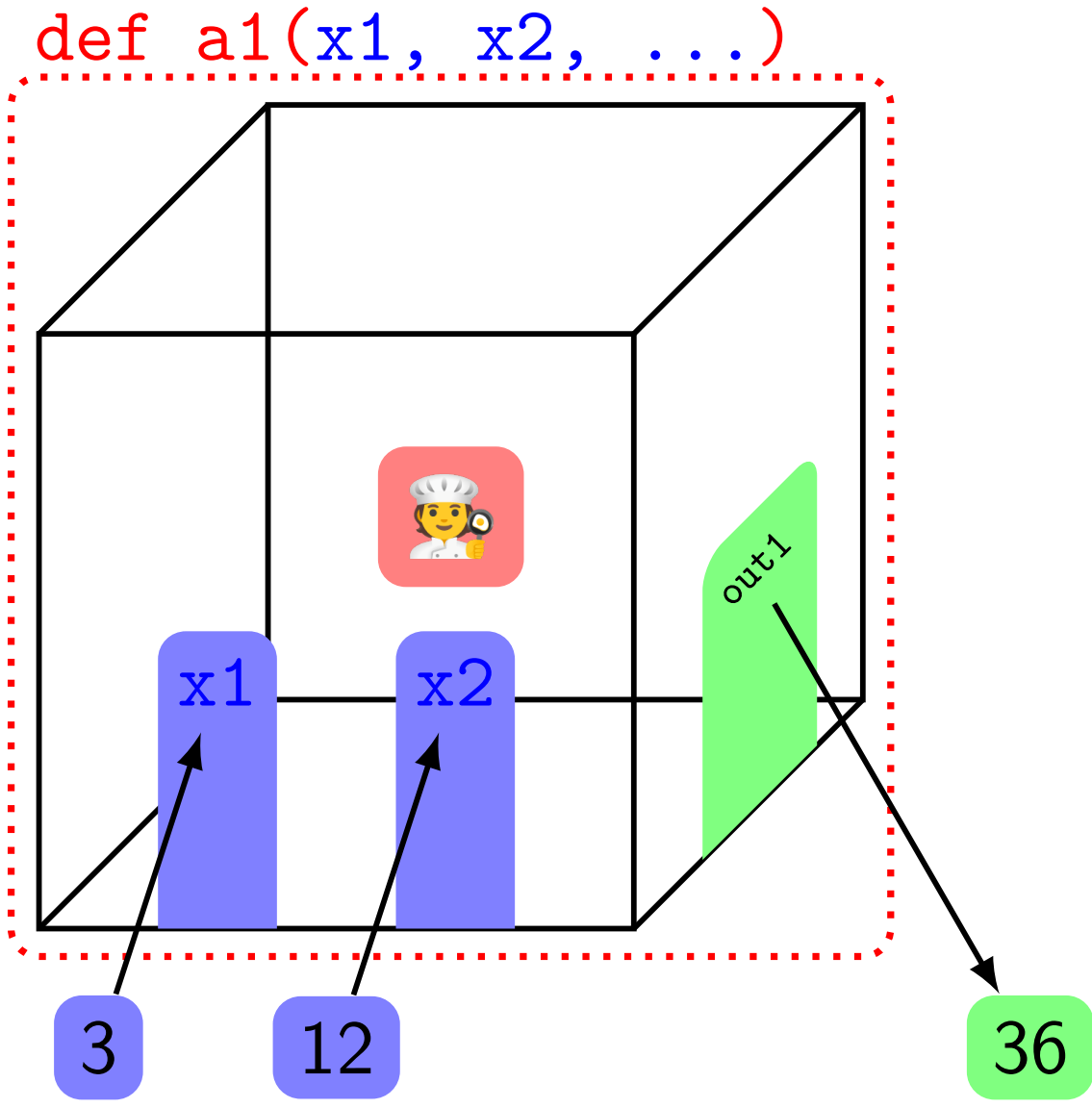
```
def summiere(x1, x2):
    summe = x1 + x2

summiere(3, 5)
print(summe)
```

Funktioniert dieser Code?
Wenn ja, was gibt er aus? Wenn nein, weshalb nicht?

Funktionen

Grundidee



Funktionen: Mit `return`

```
def summiere(x1, x2):
    summe = x1 + x2
    return summe



res = summiere(3, 5)

print(res)
```

Funktionen: Mit `return`

```
def summiere(x1, x2):
    summe = x1 + x2
    return summe

res = summiere(3, 5)

print(res)
```

Funktionen: Mit `return`

```
def summiere(x1, x2):
    summe = x1 + x2
    return summe

res = summiere(3, 5)

print(res)
```

Wert (8) an das Hauptprogramm zurückgeben...

...*und* Wert in Variable speichern, z.B. `res`

Aufträge

Im **Skript**:

- ↑ Aufgaben 1.4 - 1.7 (Abgabe auf Moodle)
- Aufgabe 1.8

Achtung

- Bitte Skript zuerst von Moodle herunterladen (Programmierkurs, Kapitel 7, „Unterlagen“)
- Codes **immer** zuerst in TigerJython ausprobieren, erst dann auf Moodle!

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:
  - 🍞 🧀 🍗 Gewisse Inputs, bzw. **Parameter** können akzeptiert werden (müssen aber nicht)

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:
  - 🍞 🧀 🍗 Gewisse Inputs, bzw. **Parameter** können akzeptiert werden (müssen aber nicht)
  - 🍔 Gewisse Outputs, bzw. **Return-Werte** können zurückgeben werden (müssen aber nicht)

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:
  - 🍞 🧀 🍗 Gewisse Inputs, bzw. **Parameter** können akzeptiert werden (müssen aber nicht)
  - 🍔 Gewisse Outputs, bzw. **Return-Werte** können zurückgeben werden (müssen aber nicht)
- 👨‍🍳 ...die wie ein **Koch** funktionieren (**Modularität**):

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:
  - 🍞 🧀 🍗 Gewisse Inputs, bzw. **Parameter** können akzeptiert werden (müssen aber nicht)
  - 🍔 Gewisse Outputs, bzw. **Return-Werte** können zurückgeben werden (müssen aber nicht)
- 👨‍🍳 ...die wie ein **Koch** funktionieren (**Modularität**):
  - Ein Koch schneidet alle Gemüse, ...

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:
  - 🍞 🧀 🍗 Gewisse Inputs, bzw. **Parameter** können akzeptiert werden (müssen aber nicht)
  - 🍔 Gewisse Outputs, bzw. **Return-Werte** können zurückgeben werden (müssen aber nicht)
- 👨‍🍳 ...die wie ein **Koch** funktionieren (**Modularität**):
  - Ein Koch schneidet alle Gemüse, ...
  - ... ein Koch grilliert die Gemüse, ...

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:
  - 🍞 🧀 🍗 Gewisse Inputs, bzw. **Parameter** können akzeptiert werden (müssen aber nicht)
  - 🍔 Gewisse Outputs, bzw. **Return-Werte** können zurückgeben werden (müssen aber nicht)
- 👨‍🍳 ...die wie ein **Koch** funktionieren (**Modularität**):
  - Ein Koch schneidet alle Gemüse, ...
  - ... ein Koch grilliert die Gemüse, ...
  - ... ein Koch bereitet die Teller schön zu, ...

Fazit

- 👍 Vorteil von `return`-Werten: man kann nun das Resultat einer Funktion in einer Variable speichern und **weiterverwenden**
- 👍 Vorteil von Funktionen: Code wird **modular**
- Funktionen sind Definitionen...
- 📋 ...die wie ein **Kochrezept** funktionieren:
  - 🍞 🧀 🍗 Gewisse Inputs, bzw. **Parameter** können akzeptiert werden (müssen aber nicht)
  - 🍔 Gewisse Outputs, bzw. **Return-Werte** können zurückgeben werden (müssen aber nicht)
- 👨‍🍳 ...die wie ein **Koch** funktionieren (**Modularität**):
  - Ein Koch schneidet alle Gemüse, ...
  - ... ein Koch grilliert die Gemüse, ...
  - ... ein Koch bereitet die Teller schön zu, ...
  - etc. (jeder Koch ist eine `def`)