# Schleifen

- Kommt oft vor, dass man die gleichen Schritte mehrfach durchführen muss
- Eine Lösung: Jeden Schritt einzeln aufschreiben
- Problem 1: Dauert (extrem!) lange
- Problem 2: Ist sehr unübersichtlich
- Problem 3: Man wird sich vertippen!
- Andere Lösung: Schleifen nutzen
- Dauert nicht viel länger als den eigentlich Schritt aufzuschreiben
- Ist kurz und (meistens) übersichtlich
- Die Gefahr des Vertippens ist wesentlich geringer

# Beispiel

- Programm, welches jede Person begrüßt & diese Personen in eine Liste schreibt

```
people = ["Lea", "Klaus", "Fatma"]
greeted = []

print("Hallo", people[0])
greeted.append(people[0])

print("Hallo", people[1])
greeted.append(people[1])

print("Hallo", people[2])
greeted.append(people[2])

print("Heute habe ich folgende Personen gegrüßt:", greeted)
```

# Beispiel – aber mit 

- Programm, welches jede Person begrüßt

```
people = ["Lea", "Klaus", "Fatma"]
greeted = []

for elem in people:
    print("Hello", elem)
    greeted.append(elem)

print("Heute habe ich folgende Personen gegrüßt:", greeted)
```

# 2 verschiedene Schleifenarten

**while-Schleife**: Solange Bedingung erfüllt ist, führe Anweisungen aus

- Bedingung: Gleiche Regeln wie bei if-Anweisungen
- Wie mehrere if-Anweisungen hintereinander
- Wichtig: Die Bedingung muss irgendwann auch nicht mehr wahr sein
  - Sonst habt ihr eine Endlosschleife…

**for-Schleife**: Für alle Werte einer Liste (o.Ä.), führe Anweisungen aus

- Zähler (Laufvariable, meist i) zählt hoch
  - Und erhöht sich bei jedem Durchlauf (von alleine!)
- Iterieren über alle Elemente einer Liste

# while-Schleifen

- Syntax:

while Bedingung:

    Anweisung 1
    Anweisung 2
    …

- Bedingung: Ein boolscher Wert (oder ein Ausdruck, der einen boolschen Wert ausgibt)
  - Muss zu Beginn wahr sein, sonst wird die Schleife ignoriert!
- Anweisung x: Befehle, die ausgeführt werden
  - Alle Anweisungen zusammen bilden einen Schleifendurchlauf

# Beispiel zu while-Schleifen

- Gebe alle Zahlen von 1 bis 5 aus

```
i = 1
while i < 6:
    print(i)
    i = i + 1


# Ausgabe
# 1. Durchlauf: 1
# 2. Durchlauf: 2
# 3. Durchlauf: 3
# 4. Durchlauf: 4
# 5. Durchlauf: 5
# Ende! Es gibt keinen 6. Durchlauf. Warum?
```

# for-Schleifen

- Syntax:

for Laufvariable in Iterable:

    Anweisung 1
    Anweisung 2
    …

- Laufvariable: Ein Variablenname, der in den Anweisung genutzt wird
  - Verweis auf das Element des aktuellen Schleifendurchlaufs
- Iterable: Eine Menge von Werten (kann Liste sein) über die iteriert wird
- Anweisung x: Befehle, die ausgeführt werden
  - Alle Anweisungen zusammen bilden einen Schleifendurchlauf

# Beispiel zu for-Schleifen

- Gebe alle Zahlen von 1 bis 5 aus

```
for i in [1, 2, 3, 4, 5]:
    print(i)
# Ausgabe, siehe Beispiel zu while-Schleifen
```

- Moment mal: Das spart mir ja gar nicht soviel Arbeit!
  - Alle Zahlen, die ausgegeben werden sollen, müssen per Hand in die Liste geschrieben werden
- Der Beste Freund von for-Schleifen: Die range()-Funktion
- Erzeugt eine Liste von Ganzzahlen bis zu einem Endwert (z.B. 5)

# Die range()-Funktion

range(*start,* stop*, step*)

- *start* und *step* sind optional, daher ignorieren wir sie fürs erste
- Der Parameter stop gibt den exklusiven Endwert an, d.h. bei welcher Ganzzahl meine Liste endet
  - Beispiel: range(6) = [0, 1, 2, 3, 4, 5] # Achtung: technisch gesehen keine Liste!
- Der Parameter *start* gibt den Startwert an (Default 0), d.h. bei welcher Ganzzahl meine Liste startet
  - Beispiel: range(1, 6) = [1, 2, 3, 4, 5] # Achtung: technisch gesehen keine Liste!
- Der Parameter *step* gibt den Abstand zwischen 2 benachbarten Zahlen an
  - Beispiel: range(1, 6, 2) = [1, 3, 5] # Achtung: technisch gesehen keine Liste!

# Übungen

- Siehe 09 – Übung.py

# Hausaufgaben

- Siehe 09 – Hausi.py

- BONUS) Wenn ihr eine zusätzliche Herausforderung sucht:

  09 – Praxisbeispiel.py