Die `for`-Schleife in Python

Einführung

Die `for`-Schleife ist eines der grundlegenden Konstrukte in Python für die Iteration. Im Gegensatz zu anderen Programmiersprachen, die auf Indizes für Iterationen basieren, ermöglicht Python die direkte Iteration über die Elemente einer Sequenz oder eines anderen iterierbaren Objekts.

Grundlegende Syntax

```
for element in iterierbar:
    # Anweisung(en), die für jedes Element ausgeführt werden
```

Durchlaufen einer Liste

Listen sind die häufigsten Sequenzen, die mit `for`-Schleifen durchlaufen werden.

Beispiel

```
fruechte = ["Apfel", "Banane", "Kirsche"]
for frucht in fruechte:
    print(frucht)
```

Verwendung von `range()`

Die `range()`-Funktion wird oft verwendet, um eine Schleife eine bestimmte Anzahl von Malen zu wiederholen.

Beispiel

```
for i in range(5):
    print(i)  # Gibt Zahlen von 0 bis 4 aus
```

Durchlaufen eines Strings

Strings können auch Zeichen für Zeichen durchlaufen werden.

Beispiel

```
for zeichen in "Hallo":
    print(zeichen)
```

Durchlaufen eines Dictionaries

Beim Durchlaufen eines Dictionaries können Schlüssel, Werte oder beides durchlaufen werden.

Beispiel

```
person = {"name": "Anna", "alter": 30}
for schlüssel in person:
    print(schlüssel)  # Gibt die Schlüssel aus

for schlüssel, wert in person.items():
    print(schlüssel, wert)  # Gibt Schlüssel und Werte aus
```

Verschachtelte Schleifen

Eine `for`-Schleife kann innerhalb einer anderen `for`-Schleife verwendet werden, um mehrdimensionale Strukturen zu durchlaufen oder andere komplexe Iterationsszenarien zu handhaben.

Beispiel

```
for i in range(3):
    for j in range(2):
        print(f"i = {i}, j = {j}")
```

Kombination mit `if`-Anweisungen

`for`-Schleifen können mit `if`-Anweisungen kombiniert werden, um bedingte Logik innerhalb der Iteration zu implementieren.

Beispiel

```
zahlen = [1, 2, 3, 4, 5]
for zahl in zahlen:
    if zahl % 2 == 0:
        print(f"{zahl} ist gerade")
    else:
        print(f"{zahl} ist ungerade")
```

Die Verwendung von `continue`, `break` und `else` in Schleifen in Python

Einführung

In Python ermöglichen die Schlüsselwörter `continue`, `break` und `else` eine feinere Kontrolle über das Verhalten von Schleifen (`for` und `while`). Sie werden verwendet, um den Ablauf innerhalb von Schleifen basierend auf Bedingungen zu steuern.

`continue`

Das Schlüsselwort `continue` wird verwendet, um die aktuelle Iteration der Schleife zu beenden und mit der nächsten Iteration fortzufahren.

Beispiel

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

In diesem Beispiel wird die Zahl 2 nicht gedruckt, da `continue` die Schleife zur nächsten Iteration springen lässt, sobald `i` gleich 2 ist.

`break`

Das Schlüsselwort `break` wird verwendet, um eine Schleife vorzeitig zu beenden, unabhängig davon, ob die Schleifenbedingung noch `True` ist oder nicht.

Beispiel

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

Hier wird die Schleife beendet, sobald `i` gleich 3 ist. Die Zahlen 0, 1 und 2 werden gedruckt, aber 3 und 4 nicht.

`else` in Schleifen

Das `else`-Schlüsselwort kann in Schleifen verwendet werden, um einen Codeblock auszuführen, nachdem die Schleife normal beendet wurde, d.h., ohne durch ein `break` beendet zu werden.

Beispiel

```
for i in range(5):
    print(i)
else:
    print("Schleife normal beendet")
```

Der `else`-Block wird ausgeführt, nachdem die `for`-Schleife alle Iterationen durchlaufen hat. Wird die Schleife jedoch mit einem `break` beendet, wird der `else`-Block nicht ausgeführt.

Beispiel mit `break` und `else`

```
for i in range(5):
    if i == 3:
        break
    print(i)
else:
    print("Schleife durch 'break' beendet")
```