Der While-Loop

Der `while`-Loop ist eine grundlegende Kontrollstruktur in Python, die es ermöglicht, einen Block von Anweisungen so lange zu wiederholen, wie eine gegebene Bedingung wahr (`True`) ist. Er ist besonders nützlich, wenn die Anzahl der Wiederholungen nicht im Voraus bekannt ist.

Grundstruktur

Die grundlegende Syntax eines `while`-Loops in Python sieht folgendermaßen aus:

```
while Bedingung:
    # Anweisungen, die wiederholt ausgeführt werden
```

- **Bedingung**: Ein Ausdruck, der bei jeder Iteration ausgewertet wird. Der Loop wird fortgesetzt, solange dieser Ausdruck `True` ergibt.
- **Anweisungen**: Der Codeblock innerhalb des Loops, der bei jeder Iteration ausgeführt wird, solange die Bedingung wahr ist.

Beispiel

Einfaches Beispiel

```
zahl = 0
while zahl < 5:
    print(zahl)
    zahl += 1
```

Dieser Loop druckt die Zahlen von 0 bis 4. Die Variable `zahl` wird bei jeder Iteration um 1 erhöht, und der Loop wird beendet, sobald `zahl` nicht mehr kleiner als 5 ist.

Verwendung mit einer Break-Anweisung

Die `break`-Anweisung kann verwendet werden, um einen `while`-Loop vorzeitig zu beenden, auch wenn die ursprüngliche Bedingung noch wahr ist.

```
zahl = 0
while True:  # Unendlicher Loop
    if zahl == 5:
        break  # Beendet den Loop, wenn zahl gleich 5 ist
    print(zahl)
    zahl += 1
```

Dieses Beispiel verwendet einen unendlichen Loop (`while True`) und beendet ihn mit einer `break`-Anweisung, sobald `zahl` gleich 5 ist.

Verwendung mit einer Continue-Anweisung

Die `continue`-Anweisung überspringt den restlichen Code innerhalb des Loops für die aktuelle Iteration und fährt mit der nächsten Iteration fort.

```
zahl = 0
while zahl < 10:
    zahl += 1
    if zahl % 2 == 0:
        continue  # Überspringt den Rest des Codes im Loop für gerade Zahlen
    print(zahl)
```

In diesem Beispiel werden nur ungerade Zahlen gedruckt, da `continue` alle geraden Zahlen überspringt.

Tipps für die Verwendung des While-Loops

- Stellen Sie sicher, dass die Bedingung des Loops irgendwann `False` wird, um eine Endlosschleife zu vermeiden.
- Verwenden Sie `break` und `continue` mit Bedacht, um die Lesbarkeit Ihres Codes zu erhalten und unbeabsichtigte Endlosschleifen zu vermeiden.
- `while`-Loops sind ideal für Situationen, in denen die Anzahl der Iterationen nicht bekannt ist oder von Bedingungen abhängt, die sich während der Ausführung des Loops ändern.