Einführung in Python: Kontrollstrukturen

Ludwig Ettner

11. Juli 2024

Übersicht

Boolesche Ausdrücke

- Boolesche Ausdrücke sind Ausdrücke, die entweder wahr (True) oder falsch (False) sind.
- Sie werden häufig verwendet, um Bedingungen auszudrücken oder Entscheidungen zu treffen.

Logische Operatoren

- **AND (und) - and**: Wahr, wenn beide Operanden wahr sind. Beispiel: `True and True` ergibt True.
- **OR (oder) - or**: Wahr, wenn mindestens ein Operand wahr ist. Beispiel: `True or False` ergibt True.
- **NOT (nicht) - not**: Kehrt den Wert des Operanden um. Beispiel: `not True` ergibt False.

Vergleichsoperatoren

- Größer als (>)
- Kleiner als (<)
- Größer oder gleich (>=)
- Kleiner oder gleich (<=)
- Gleich (==)
- Ungleich (!=)

Bedingungen

- Bedingungen ermoeglichen es uns, Codebloecke nur dann auszufuehren, wenn bestimmte Bedingungen erfuellt sind.
- **if-Bedingung**:

```
1 if Bedingung:
2     # Code wird ausgef\"uhrt, wenn die Bedingung
3     # wahr ist
```

- **else-Bedingung**:

```
1 else:
2     # Code wird ausgef\"uhrt, wenn die Bedingung
3     # falsch ist
```

Erweiterte Bedingungen mit elif

- **elif-Bedingung**:

```
if Bedingung1:
    # Code wird ausgefuehrt, wenn Bedingung1
    # wahr ist
elif Bedingung2:
    # Code wird ausgefuehrt, wenn Bedingung1
    # falsch, aber Bedingung2 wahr ist
else:
    # Code wird ausgefuehrt, wenn keine der
    # Bedingungen wahr ist
```

while-Schleife

- Eine while-Schleife wird verwendet, um einen Codeblock solange zu wiederholen, wie eine bestimmte Bedingung wahr ist.
- **Syntax**:

```
while Bedingung:
    # Code wird wiederholt ausgefueuhrt,
    # solange die Bedingung wahr ist
```

while-Schleife Beispiel

```
1 counter = 0
2 while counter < 5:
3     print("Counter:", counter)
4     counter += 1
```

- Ausgabe:

```
1 Counter: 0
2 Counter: 1
3 Counter: 2
4 Counter: 3
5 Counter: 4
```

for-Schleife

- Eine for-Schleife wird verwendet, um einen Codeblock für jedes Element in einer Sequenz auszuführen.
- **Syntax**:

```
1 for Element in Sequenz:
2     # Code wird fuer jedes Element in
3     # der Sequenz ausgefuehrt
```

for-Schleife Beispiel

```
1 fruits = ["Apple", "Banana", "Orange"]
2 for fruit in fruits:
3     print(fruit)
```

- Ausgabe:

```
1 Apple
2 Banana
3 Orange
```

Erweiterte Funktionen der for-Schleife

- For-Schleifen können mit Listen und Dictionaries verwendet werden, um komplexe Datenstrukturen zu durchlaufen.

For-Schleife mit Listen

- List Comprehensions: Eine kompakte Möglichkeit, Listen zu erstellen.

```
squares = [x**2 for x in range(10)]
print(squares)
```

- Ausgabe:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

For-Schleife mit Dictionaries

- Dictionaries durchlaufen:

```
student_grades = {"Alice": 85,
                  "Bob": 72,
                  "Charlie": 90}
for student, grade in student_grades.items():
    print(f"{student}: {grade}")
```

- Ausgabe:

```
Alice: 85
Bob: 72
Charlie: 90
```

Nützliche Funktionen

- enumerate(): Fügt den Elementen in der Schleife einen Zähler hinzu.

```
fruits = ["Apple", "Banana", "Orange"]
for index, fruit in enumerate(fruits):
    print(index, fruit)
```

- zip(): Ermöglicht das Durchlaufen mehrerer Listen gleichzeitig.

```
names = ["Alice", "Bob", "Charlie"]
grades = [85, 72, 90]
for name, grade in zip(names, grades):
    print(f"{name}: {grade}")
```

Fazit

- Kontrollstrukturen sind essenziell, um den Programmfluss zu steuern.
- Boolesche Ausdrücke, Bedingungen und Schleifen bieten vielseitige Möglichkeiten.
- Erweiterte Funktionen der for-Schleife ermöglichen effizientes Arbeiten mit Listen und Dictionaries.