LU04c - List Comprehensions

List Comprehensions bieten eine einfache Möglichkeit, Listen zu erstellen. Sie ermöglichen es, Ausdrücke auf eine Weise zu schreiben, die sich sehr natürlich und näher an der mathematischen Schreibweise anfühlt.

Hauptverwendungszweck von List Comprehensions

List Comprehensions werden in Python vorwiegend zur Transformation von Daten verwendet. Sie sind besonders nützlich, wenn man eine bestehende Liste (oder ein anderes iterierbares Objekt) nehmen und jedes ihrer Elemente auf eine bestimmte Weise transformieren möchte. Außerdem kann man mit ihnen Elemente basierend auf bestimmten Kriterien filtern.

Die Syntax

```
newlist = [expression for item in iterable if condition == True]

#Identisch mit:

newlist = []
for item in interable:
    if condition:
         newlist.append(expression)
```

Teile einer List Comprehension

Condition

Ein optionaler Filter, um nur bestimmte Elemente einzuschließen.

```
cars = ['audi', 'vw', 'seat', 'skoda', 'tesla']
newlist = [car for car in cars if car != 'tesla']
#newlist = ['audi', 'vw', 'seat', 'skoda']
```

Die Bedingung `if x != 'tesla'` liefert `True` für alle Elemente außer `tesla`, so dass die neue Liste alle Autos außer `Tesla` enthält.

Die Bedingung ist optional und kann weggelassen werden:

```
newlist = [x for x in cars]
```

Iterable

Das Iterable kann ein beliebiges iterierbares Objekt sein, z. B. eine Liste, ein Tupel, ein Set usw.

1. **Beispiel mit einer Liste**:

```
squared = [x**2 for x in [1, 2, 3, 4]]
# Ergebnis: [1, 4, 9, 16]

numbers = [1, 2, 3, 4]
squared = [x**2 for x in numbers]
# Ergebnis: [1, 4, 9, 16]
```

2. **Beispiel mit einem Tupel**:

```
doubled = [x * 2 for x in (1, 2, 3, 4)]
# Ergebnis: [2, 4, 6, 8]
```

3. **Beispiel mit range()**:

```
newlist = [x for x in range(10)]
# Ergebnis: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Expression

Der aktuelle Gegenstand der Iteration, aber auch das Ergebnis, das vor seinem Einzug in die neue Liste manipuliert werden kann.

1. **Beispiel mathematische Operation**:

```
#Expression = x * 2
doubled = [x * 2 for x in range(5)]
# Ergebnis: [0, 2, 4, 6, 8]
```

2. **Beispiel mit Funktionsaufruf**:

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
newlist = [car.upper() for car in cars]
print(newlist)
```

Item

Das aktuelle Element, das in der Iteration verarbeitet wird. Der Name des Item wird im `for`-Konstrukt

definiert: `for item in list`

Beispiel

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
newlist = [item.upper() for item in cars]
print(newlist)
```

Weitere Beispiele

1. **Mit einem Iterable eines anderen Typs**:

```
letters = [letter for letter in 'Hallo Welt']
# Ergebnis: ['H', 'a', 'l', 'l', 'o', ' ', 'W', 'e', 'l', 't']
```

2. **Nested List Comprehensions**:

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flattened = [num for row in matrix for num in row]
# Ergebnis: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

List Comprehensions sind ein mächtiges Werkzeug in Python, das den Code sauberer und verständlicher machen kann. Sie ermöglichen die schnelle Transformation von Daten und unterstützen dabei, den Code näher an die menschliche Logik und mathematische Notation zu bringen.

M323-LU04