Generator Expressions und Yield-Funktionen in Python

Einführung

Generator Expressions und Yield-Funktionen sind leistungsstarke Werkzeuge in Python, die es ermöglichen, effizient Sequenzen von Elementen zu erzeugen, ohne die gesamte Sequenz im Speicher zu halten. Sie sind besonders nützlich, wenn große Datenmengen schrittweise verarbeitet werden müssen oder wenn die vollständige Sequenz nie benötigt wird.

Generator Expressions

Generator Expressions sind eine kompakte Möglichkeit, Generator-Objekte zu erstellen. Sie ähneln List Comprehensions, erzeugen jedoch ein Generator-Objekt anstelle einer Liste.

Syntax

```
generator_expression = (ausdruck for element in iterable)
```

Beispiel

```
generator = (x ** 2 for x in range(5))
for element in generator:
    print(element)
```

Yield-Funktionen

Yield-Funktionen sind Funktionen, die das Schlüsselwort `yield` verwenden, um Werte schrittweise zu generieren. Jedes Mal, wenn die `yield`-Anweisung erreicht wird, wird der aktuelle Zustand der Funktion gespeichert, und der Wert wird zurückgegeben. Die Funktion wird fortgesetzt, wenn der nächste Wert benötigt wird.

Syntax

```
def meine_yield_funktion():
    for element in iterable:
        yield ausdruck
```

Beispiel

```
def quadrat_generator(n):
    for i in range(n):
```

```
        yield i ** 2

for element in quadrat_generator(5):
    print(element)
```

Unterschiede zwischen Generator Expressions und Yield-Funktionen

- Generator Expressions sind eine kompakte, inline-Definition von Generatoren, während Yield-Funktionen separate Funktionen sind.
- Generator Expressions werden in geschweiften Klammern `()` definiert, während Yield-Funktionen das Schlüsselwort `yield` verwenden, um Werte zu erzeugen.

Verwendungszwecke

- Generator Expressions und Yield-Funktionen sind nützlich, wenn große Datenmengen schrittweise verarbeitet werden müssen, um Speicher zu sparen.
- Sie sind ideal, wenn die vollständige Sequenz nie benötigt wird, sondern nur ein Element nach dem anderen.