LU04h - Generatoren

Generatoren sind eine einfache Möglichkeit, Iteratoren in Python zu erstellen. Sie ermöglichen es, durch eine Sequenz von Werten zu iterieren, ohne die gesamte Sequenz im Speicher zu speichern. Dies wird durch die Verwendung des Schlüsselworts `yield` anstelle von `return` in einer Funktion erreicht.

Was sind Generatoren?

Ein Generator ist ein spezieller Typ von Iterator, der mit einer Funktion und dem Schlüsselwort `yield` erstellt wird. Im Gegensatz zu normalen Funktionen, die einen Wert zurückgeben und dann beendet werden, „pausieren“ Generatoren ihren Zustand und setzen die Ausführung fort, wenn der nächste Wert benötigt wird.

Syntax

Die Erstellung eines Generators ist ähnlich wie die einer normalen Funktion, aber anstelle des `return`-Schlüsselworts verwenden wir `yield`:

```
def my_generator():
    yield 1
    yield 2
    yield 3
```

Beispiele

Einfacher Generator

Ein einfacher Generator, der die Zahlen 1 bis 3 zurückgibt:

```
def simple_generator():
    yield 1
    yield 2
    yield 3

gen = simple_generator()
print(next(gen))  # Output: 1
print(next(gen))  # Output: 2
print(next(gen))  # Output: 3
```

Generator mit Schleife

Ein Generator, der die Quadrate der Zahlen von 1 bis n zurückgibt:

```
def square_numbers(n):
    for i in range(1, n+1):
        yield i * i

for square in square_numbers(5):
    print(square)
# Output: 1, 4, 9, 16, 25
```

Generator Expression

Generator Expressions sind eine kompaktere Art, Generatoren zu erstellen, ähnlich wie List Comprehensions:

```
squared = (x*x for x in range(6))
print(next(squared))  # Output: 0
print(next(squared))  # Output: 1
```

Vorteile von Generatoren

- **Speichereffizienz**: Generatoren speichern nicht alle Werte im Speicher, sondern generieren sie „on-the-fly“. Dies ist besonders nützlich, wenn mit großen Datenmengen gearbeitet wird.
- **Einfachheit**: Generatoren sind einfacher zu implementieren als benutzerdefinierte Iteratoren.
- **Vielseitigkeit**: Generatoren können für jede Art von Datenquelle verwendet werden, nicht nur für Listen oder Sequenzen.

Verständnis des ''yield''-Schlüsselworts

Das Schlüsselwort `yield` in Python wird verwendet, um einen Wert aus einer Funktion zurückzugeben und den aktuellen Zustand der Funktion zu speichern. Wenn die Funktion das nächste Mal aufgerufen wird, wird sie von der Stelle fortgesetzt, an der sie zuletzt angehalten wurde, und nicht von Anfang an. Dies ermöglicht es der Funktion, Werte „on-the-fly“ zu generieren, ohne den gesamten Datensatz im Speicher zu halten.

M323-LU04