Lambda-Funktionen in Python

Einführung

Lambda-Funktionen in Python sind eine kompakte Methode, um anonyme Funktionen zu erstellen. Sie sind besonders nützlich für Operationen, die kleine Funktionen benötigen, wie beim Sortieren oder Filtern von Daten.

Syntax

Die allgemeine Syntax einer Lambda-Funktion ist:

```
lambda argumente: ausdruck
```

Lambda-Funktionen können mehrere Argumente annehmen, enthalten aber nur einen Ausdruck.

Beispiele

Ein einfaches Beispiel

```
quadrat = lambda x: x ** 2
print(quadrat(5))  # Ausgabe: 25
```

Mehrere Argumente

```
addiere = lambda x, y: x + y
print(addiere(5, 3))  # Ausgabe: 8
```

Verwendung mit eingebauten Funktionen

Lambda-Funktionen werden häufig mit Funktionen wie `map()`, `filter()` und `sorted()` verwendet.

Verwendung mit `filter()`

```
zahlen = [1, 2, 3, 4, 5, 6]
gerade_zahlen = list(filter(lambda x: x % 2 == 0, zahlen))
print(gerade_zahlen)  # Ausgabe: [2, 4, 6]
```

Verwendung mit `map()`

```
zahlen = [1, 2, 3, 4, 5]
quadrate = list(map(lambda x: x ** 2, zahlen))
print(quadrate)  # Ausgabe: [1, 4, 9, 16, 25]
```

Verwendung mit `sorted()`

```
punkte = [(1, 'eins'), (4, 'vier'), (2, 'zwei'), (3, 'drei')]
sortiert_nach_zahl = sorted(punkte, key=lambda x: x[0])
print(sortiert_nach_zahl)  # Ausgabe: [(1, 'eins'), (2, 'zwei'), (3, 'drei'), (4,
'vier')]
```

Vorteile

- **Kompaktheit**: Lambda-Funktionen ermöglichen es, kleine Funktionen in einer einzigen Zeile zu schreiben.
- **Anonymität**: Lambda-Funktionen benötigen keinen Namen, was sie ideal für die Verwendung als Argumente in höheren Funktionen macht.

Einschränkungen

- **Beschränkung auf einen Ausdruck**: Lambda-Funktionen können nur einen Ausdruck enthalten und sind daher in ihrer Komplexität begrenzt.
- **Lesbarkeit**: Bei übermäßiger oder unsachgemäßer Verwendung können Lambda-Funktionen den Code schwer lesbar machen.