Python Tutorial: Operator-Überladung

Einführung

In Python können Sie das Verhalten von Standardoperatoren für Ihre Klassen ändern, indem Sie spezielle Methoden implementieren. Diese Methoden werden automatisch aufgerufen, wenn Sie Operatoren wie Addition (+), Subtraktion (-), Multiplikation (*) usw. auf die Instanzen Ihrer Klasse anwenden.

Grundlegende Operatoren und ihre magischen Methoden

Addition: `__add__(self, other)`

Definiert das Verhalten der Addition (+) für die Instanz der Klasse.

```
class Punkt:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Punkt(self.x + other.x, self.y + other.y)
```

Subtraktion: `__sub__(self, other)`

Definiert das Verhalten der Subtraktion (-).

```
class Punkt:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __sub__(self, other):
        return Punkt(self.x - other.x, self.y - other.y)
```

Multiplikation: `__mul__(self, other)`

Definiert das Verhalten der Multiplikation (*).

```
class Punkt:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __mul__(self, other):
        return Punkt(self.x * other, self.y * other)
```

Weitere Operatoren

- Division: `__truediv__(self, other)` für `/`
- Gleichheit: `__eq__(self, other)` für `==`
- Ungleichheit: `__ne__(self, other)` für `!=`
- Und viele mehr, abhängig von den Anforderungen Ihrer Klasse.

Beispiel: Eine Klasse mit mehreren Operator-Überladungen

```
class Bruch:
    def __init__(self, zaehler, nenner):
        self.zaehler = zaehler
        self.nenner = nenner

    def __str__(self):
        return f"{self.zaehler}/{self.nenner}"

    def __add__(self, other):
        neuer_zaehler = self.zaehler * other.nenner + self.nenner * other.zaehler
        neuer_nenner = self.nenner * other.nenner
        return Bruch(neuer_zaehler, neuer_nenner)

    # Weitere Operator-Überladungen hier einfügen

bruch1 = Bruch(1, 2)
bruch2 = Bruch(3, 4)
ergebnis = bruch1 + bruch2
print(ergebnis)  # Gibt 10/8 oder 5/4 nach Kürzung aus
```

Zusammenfassung

Die Operator-Überladung in Python ermöglicht eine intuitive Nutzung benutzerdefinierter Klassen in arithmetischen und anderen Operationen. Durch die Implementierung spezieller magischer Methoden können Sie die Interaktionen zwischen Instanzen Ihrer Klassen anpassen und verbessern, was zu einem klareren und ausdrucksstärkeren Code führt.