Vererbung

Einführung

Vererbung ermöglicht es einer Klasse (der abgeleiteten Klasse) die Attribute und Methoden einer anderen Klasse (der Basisklasse) zu erben. Python unterstützt sowohl Einzel- als auch Mehrfachvererbung, was bedeutet, dass eine Klasse von mehr als einer Basisklasse erben kann.

Grundlagen der Vererbung

Definieren einer Basisklasse

Eine Basisklasse ist die Klasse, von der Eigenschaften geerbt werden.

```
class Fahrzeug:
    def __init__(self, marke, modell):
        self.marke = marke
        self.modell = modell

    def zeige_details(self):
        print(f"Marke: {self.marke}, Modell: {self.modell}")
```

Erstellen einer abgeleiteten Klasse

Eine abgeleitete Klasse erbt Eigenschaften von der Basisklasse. Sie kann auch eigene Eigenschaften und Methoden definieren oder Methoden der Basisklasse überschreiben.

```
class Auto(Fahrzeug):  # Erbt von Fahrzeug
    def __init__(self, marke, modell, ps):
        super().__init__(marke, modell)  # Ruft den Konstruktor der Basisklasse
auf
        self.ps = ps

    def zeige_details(self):
        super().zeige_details()  # Ruft die Methode der Basisklasse auf
        print(f"PS: {self.ps}")
```

Verwendung der Vererbung

Instanzen der abgeleiteten Klasse haben Zugriff auf die Methoden und Attribute der Basisklasse sowie auf alle überschriebenen oder neu definierten Eigenschaften.

```
mein_auto = Auto("Tesla", "Model S", 691)
mein_auto.zeige_details()
```

Mehrfachvererbung

Python unterstützt Mehrfachvererbung, was bedeutet, dass eine Klasse von mehreren Basisklassen erben kann.

```
class Elektrisch:
    def __init__(self, batterie):
        self.batterie = batterie

    def zeige_batterie(self):
        print(f"Batteriekapazität: {self.batterie} kWh")

class Elektroauto(Auto, Elektrisch):  # Erbt von Auto UND Elektrisch
    def __init__(self, marke, modell, ps, batterie):
        Auto.__init__(self, marke, modell, ps)
        Elektrisch.__init__(self, batterie)
```

Vererbungsprobleme lösen

Bei Mehrfachvererbung können Konflikte auftreten, wenn Basisklassen dieselben Methodennamen haben. Python löst solche Konflikte mit der Method Resolution Order (MRO), die definiert, wie Methoden aufgelöst werden.

`super()` und Method Resolution Order (MRO)

Einführung

In Python ermöglicht die `super()` Funktion den Zugriff auf die Methoden der Basisklasse von einer abgeleiteten Klasse aus, was besonders nützlich ist, um den Konstruktor der Basisklasse aufzurufen oder Methoden zu überschreiben. Die Method Resolution Order (MRO) definiert die Reihenfolge, in der Basisklassen abgesucht werden, um die angeforderte Methode zu finden.

Die `super()` Funktion

Grundlegende Nutzung

`super()` wird verwendet, um eine Methode der Basisklasse aufzurufen, ohne den Namen der Basisklasse explizit angeben zu müssen. Dies ist besonders hilfreich in der Mehrfachvererbung, da es die Wartung des Codes vereinfacht.

```
class Basis:
    def __init__(self, wert):
        self.wert = wert

class Abgeleitet(Basis):
    def __init__(self, wert):
        super().__init__(wert)  # Ruft den Konstruktor der Basisklasse auf
```

`super()` in Mehrfachvererbung

In der Mehrfachvererbung ermöglicht `super()` den Zugriff auf die nächste Klasse in der MRO.

```
class A:
    def zeige(self):
        print("A", end=" ")

class B(A):
    def zeige(self):
        super().zeige()
        print("B", end=" ")

class C(A):
    def zeige(self):
        super().zeige()
        print("C", end=" ")

class D(B, C):
    def zeige(self):
        super().zeige()
        print("D", end=" ")

d = D()
d.zeige()  # Ausgabe: A C B D
```

Method Resolution Order (MRO)

Was ist MRO?

Die Method Resolution Order (MRO) ist die Reihenfolge, in der Python sucht, um die richtige Methode oder das Attribut in der Vererbungshierarchie zu finden. Python verwendet das C3 Linearisierungs-Algorithmus, um eine eindeutige MRO für jede Klasse zu erstellen.

Anzeigen der MRO

Die MRO einer Klasse kann mit der Methode `mro()` oder dem Attribut `__mro__` angezeigt werden.

```
print(D.mro())  # Zeigt die MRO für die Klasse D
# [D, B, C, A, object]
```

Bedeutung der MRO

Die MRO ist entscheidend in der Mehrfachvererbung, da sie bestimmt, wie Konflikte zwischen den Basisklassen aufgelöst werden. Sie sorgt dafür, dass jede Methode oder jedes Attribut eindeutig einer Klasse zugeordnet wird, was Mehrdeutigkeiten vermeidet.