Properties

Einführung

Properties in Python sind spezielle Methoden, die den Zugriff auf die Attribute einer Klasse "verpacken". Sie ermöglichen es, Getter, Setter und Deleter Methoden auf eine Weise zu definieren, die den Zugriff auf die Attribute einer Klasse ähnlich dem Zugriff auf einfache Attribute erscheinen lässt.

Grundlagen der Properties

Definieren von Properties

Um eine Property in Python zu definieren, verwendet man den `@property` Dekorateur oberhalb einer Methode. Diese Methode definiert dann den Getter für die Property.

```
class Person:
    def __init__(self, name):
        self._name = name

    @property
    def name(self):
        return self._name
```

In diesem Beispiel ist `name` eine Property, und `_name` ist ein privates Attribut, das die tatsächlichen Daten speichert.

Definieren von Settern

Um den Wert einer Property zu setzen, definiert man einen Setter mit dem Dekorateur `@property_name.setter`.

```
class Person:
    def __init__(self, name):
        self._name = name

    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, value):
        if not value:
            raise ValueError("Name darf nicht leer sein.")
        self._name = value
```

Definieren von Deletern

Ein Deleter kann definiert werden, um spezielle Aktionen durchzuführen, wenn ein Attribut gelöscht wird (z.B. mit `del obj.property_name`).

```
class Person:
    def __init__(self, name):
        self._name = name

    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, value):
        if not value:
            raise ValueError("Name darf nicht leer sein.")
        self._name = value

    @name.deleter
    def name(self):
        print("Name wird gelöscht!")
        del self._name
```

Verwendung von Properties

Properties sind nützlich, um Validierungen durchzuführen, Logging zu implementieren oder Berechnungen durchzuführen, wenn ein Attribut gelesen oder geschrieben wird.

```
p = Person("Alex")
print(p.name)  # Ruft den Getter auf

p.name = "Max"  # Ruft den Setter auf, ändert den Namen zu Max

del p.name  # Ruft den Deleter auf und löscht den Namen
```

Vorteile von Properties

- **Datenkapselung**: Ermöglicht es, den direkten Zugriff auf Attribute zu kontrollieren und Validierungen oder andere Logik beim Setzen eines Attributs einzufügen.
- **Benutzerfreundlichkeit**: Properties ermöglichen es, Methoden wie normale Attribute zu verwenden, was den Zugriff intuitiv macht.
- **Kompatibilität**: Ermöglicht es, Klassenattribute, die ursprünglich öffentlich waren, in Properties umzuwandeln, ohne die öffentliche API der Klasse zu ändern.

Zusammenfassung

Properties in Python bieten eine Methode, um den Zugriff auf die Attribute einer Klasse zu steuern. Durch die Verwendung von Gettern, Settern und Deletern kann man die Datenkapselung verbessern und sicherstellen,

dass die Attribute einer Klasse auf sichere und kontrollierte Weise verwendet werden.