

# Magische Methoden

---

## Einführung

Magische Methoden in Python sind spezielle Methoden, die mit doppelten Unterstrichen (`__`) beginnen und enden. Sie ermöglichen es Entwicklern, das Verhalten von Klassen zu definieren und anzupassen, sodass die Instanzen dieser Klassen sich wie eingebaute Python-Objekte verhalten können.

Magische Methoden, auch bekannt als Dunder-Methoden (Double Underscores), ermöglichen es uns, das Verhalten von Python-Objekten zu ändern und anzupassen, sodass sie sich wie eingebaute Typen verhalten. Sie können für eine Vielzahl von Operationen überladen werden, wie z.B. arithmetische Operationen, Vergleiche, die Darstellung von Objekten und die Containerverwaltung.

## Wichtige Magische Methoden

### `__init__` und `__del__`

- `__init__(self, [...])`: Wird aufgerufen, wenn eine neue Instanz der Klasse erstellt wird. Dient zur Initialisierung der Instanz.
- `__del__(self)`: Wird aufgerufen, wenn eine Instanz gelöscht wird. Kann für Aufräumarbeiten verwendet werden.

### `__str__` und `__repr__`

- `__str__(self)`: Gibt eine informelle, lesbare Darstellung des Objekts zurück, die für Endbenutzer gedacht ist.
- `__repr__(self)`: Gibt eine offizielle Darstellung des Objekts zurück, die mehr für Entwickler gedacht ist. Idealerweise sollte sie so sein, dass `eval(repr(obj))` das Objekt rekonstruieren kann.

## Arithmetische Operationen

- `__add__(self, other)`: Definiert das Verhalten der Addition (+).
- `__sub__(self, other)`: Definiert das Verhalten der Subtraktion (-).
- Weitere Methoden wie `__mul__`, `__div__`, etc. erlauben die Überladung anderer arithmetischer Operatoren.

## Vergleichsoperationen

- `__eq__(self, other)`: Gleichheit (==).
- `__ne__(self, other)`: Ungleichheit (!=).
- `__lt__(self, other)`: Kleiner als (<).
- Weitere Methoden wie `__le__`, `__gt__`, und `__ge__` überladen die restlichen Vergleichsoperatoren.

## Containermethoden

- `__len__(self)`: Gibt die Länge des Containers zurück (für `len(obj)`).
- `__getitem__(self, key)`: Ermöglicht den Zugriff auf Elemente mit `obj[key]`.
- `__setitem__(self, key, value)`: Ermöglicht das Setzen von Elementen mit `obj[key] = value`.

- `__delitem__(self, key)`: Ermöglicht das Löschen von Elementen mit `del obj[key]`.

## Beispiel: Eine einfache Klasse mit Magischen Methoden

```
class Buch:
    def __init__(self, titel, autor):
        self.titel = titel
        self.autor = autor

    def __str__(self):
        return f"{self.titel} von {self.autor}"

    def __repr__(self):
        return f"Buch('{self.titel}', '{self.autor}')"

buch = Buch("Python 101", "Max Muster")
print(buch)  # Ruft __str__ auf
repr(buch)   # Ruft __repr__ auf
```

## Zusammenfassung

Magische Methoden in Python bieten eine leistungsstarke Möglichkeit, das Verhalten von Objekten zu definieren und anzupassen. Sie erlauben es, Klassen so zu gestalten, dass ihre Instanzen sich wie eingebaute Python-Objekte verhalten, was die Durchführung von Operationen wie Addition, Vergleich und sogar die Darstellung der Objekte in einer benutzerfreundlichen Weise ermöglicht.