Magische Attribute und Methoden in Python

Definition:

- Python-Magic-Methoden und -Attribute sind spezielle Methoden (__method__) und Attribute (__attribute__), die mit doppelten Unterstrichen beginnen und enden - z.B. __init__, __str__ , __add__, __dict__ , __bases__

- Sie wirken wie versteckte Zaubertricks, die das Verhalten von Objekten anpassen.

- Sie werden automatisch aufgerufen, wenn bestimmte Aktionen passieren, z.B. :

  Addition **+** → __add__
  Stringausgabe **print(**obj**)** → __str__
  Objektvergleich **==** → __eq__
  Länge berechnen **len(**obj**)** → __len__
  Objekt erstellen **Klasse()** → __init__

Magische Attribute und Methoden in Python

Zweck:

- Python nutzt magische Methoden im Hintergrund , damit eingebaute Funktionen wie **print()**, **len()** oder **str()** sinnvoll mit Objekten interagieren können.

- Magische Methoden ermöglichen es, eigene Klassen so zu gestalten, dass sie sich wie eingebaute Typen (z. B. int, list) verhalten.

- Dadurch lassen sich auch eigene Operatoren, Attributzugriffe und Spezialfunktionen definieren.

Häufige Magische Methoden:

Konstruktion & String-Repräsentation

__init__ : Konstruktor

__new__ : Objekt wird erzeugt

__str__ : Gibt eine verständliche, schön lesbare Beschreibung des Objekts zurück – z. B. für print(obj).

__repr__ : Gibt eine technische Darstellung zurück, nützlich für Entwickler und beim Debuggen.

Magische Attribute und Methoden in Python

Magische Attribute

__dict__ : Wörterbuch für die Attribute eines Objekts.

__name__ : Gibt den Namen einer Funktion, Klasse oder eines Moduls zurück. Nicht verfügbar bei Instanzen (also normalen Objekten).

__doc__ : Gibt den Dokumentationsstring einer Funktion, Methode, Klasse oder eines Moduls zurück – wenn vorhanden. Bei Instanzen wird – falls vorhanden – die Klassendokumentation angezeigt.

__module__ : Modul, in dem die Klasse oder Funktion definiert wurde.

__bases__ : Basisklassen einer Klasse (als Tuple).

__class__ : Klasse, zu der eine Instanz gehört.

Überladen von Operatoren

Arithmetische Operatoren: __add__, __sub__, __mul__, __truediv__, __floordiv__

Vergleichsoperatoren: __eq__, __lt__, __gt__, __ne__, usw.

Magische Attribute und Methoden in Python

Container-Magische Methoden

Zugriff und Zuweisung: __getitem__, __setitem__, __delitem__

→ Werden automatisch verwendet bei Indexzugriff, Zuweisung oder Löschung – z. B. obj[0] , obj[1] = x , del obj[2]

Länge und Umkehrung: __len__, __reversed__

→ Gesteuert über len(obj) und reversed(obj)

Fortgeschrittene Magische Methoden

Attributzugriff: __getattr__, __getattribute__, __setattr__, __delattr__

→ Wird aufgerufen, wenn auf Attribute zugegriffen, sie gesetzt oder gelöscht werden – z. B. obj.name, obj.name = x, del obj.name

__getattr__ wird nur aufgerufen, wenn das angeforderte Attribut nicht existiert oder nicht gefunden wird.

__getattribute__ wird immer aufgerufen – aber sei vorsichtig, um Endlosschleifen zu vermeiden. Lösung super().__getattribute__(attr)

Es ruft die Standard-Implementierung aus der Elternklasse (object) auf – ohne sich selbst erneut zu triggern.

Aufrufbare Objekte: __call__

→ Erlaubt, dass ein Objekt wie eine Funktion aufgerufen werden kann – z. B. obj("Hallo").
Wird z. B. bei Machine-Learning-Modellen verwendet - model(input)

Kontextmanager

Betreten und Verlassen: __enter__, __exit__

→ Erlaubt die Verwendung in with-Blöcken, z. B. with obj

Magische Attribute und Methoden in Python

Wir können alle magischen Methoden einer Klasse in Python mithilfe der eingebauten dir()-Funktion inspizieren und auflisten.

z.B. Wir können die dir()-Funktion verwenden, um die int-Klasse in Python zu untersuchen und dann eine Zusammenfassung ihrer magischen Methoden und Schlüsselfunktionen zu geben.

```
print(dir(int))
```

```
# Magische Methoden für grundlegende arithmetische Operationen sind hier mit blauer Farbe hervorgehoben
```

```
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__',
'__float__', '__floor__', '__floordiv__', '__format__', '__ge__', '__getattribute__', '__getnewargs__', '__getstate__', '__gt__',
'__hash__', '__index__', '__init__', '__init_subclass__', '__int__', '__invert__', '__le__', '__lshift__', '__lt__', '__mod__',
'__mul__', '__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduce__',
'__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__round__', '__rpow__',
'__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__',
'__subclasshook__', '__truediv__', '__trunc__', '__xor__', 'as_integer_ratio', 'bit_count', 'bit_length', 'conjugate', 'denominator',
'from_bytes', 'imag', 'is_integer', 'numerator', 'real', 'to_bytes']
```

Code-Beispiel - Magische Attribute

```
class Auto:
    """
    Eine einfache Klasse, die ein Auto repräsentiert. Sie speichert Informationen über das Modell,
    die Marke und das Herstellungsjahr des Autos.
    """


    def __init__(self, marke, modell, jahr):
        self.marke = marke
        self.modell = modell
        self.jahr = jahr

# Eine Instanz von Auto erstellen
mein_auto = Auto("Toyota", "Corolla", 2021)

# Verwendung von magischen Attributen
print("Dokumentation der Auto-Klasse:", Auto.__doc__)
print("Wörterbuch der Auto-Instanz:", mein_auto.__dict__)
print("Klass :", mein_auto.__class__)
print("Klassenname :", mein_auto.__class__.__name__)
print("Class Module:", Auto.__module__)
print("Class Elternklassen:", Auto.__bases__)
```

Code-Beispiel - Überladen von Operatoren

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

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

    def __str__(self):
        return f"Vector({self.x}, {self.y})"
        #__str__ Methode gibt dieses Objekt in einer lesbaren Form aus


v1 = Vector(2, 3)
v2 = Vector(5, 7)
print(v1 + v2)  # dies gibt ein neues Objekt zurück

# Ausgabe:  Vector(7, 10)
```

Zusammenfassung

- Magische Methoden und Attribute erweitern die objektorientierten Fähigkeiten von Python, indem sie benutzerdefinierte Verhaltensweisen für Operationen wie Arithmetik, Vergleiche und Attributverwaltung ermöglichen.
- Diese Funktionen ermöglichen die Implementierung von Operatorüberladung, Polymorphismus und mehr, was Python sehr flexibel und ausdrucksstark macht.
- Durch das Verständnis und die Nutzung von magischen Methoden und Attributen können Entwickler intuitivere und robustere Python-Klassen erstellen, die sich wie eingebaute Typen verhalten.
- Diese Mechanismen unterstützen auch fortgeschrittene Funktionen wie Kontextmanager und iterierbare Objekte, was zu saubererem und wartungsfreundlicherem Code beiträgt.
- Magische Methoden und Attribute sind grundlegend für das Beherrschen von fortgeschrittenen Funktionen in Python und dienen als **Brücke zwischen Anfängerkodierung und komplexeren Software-Designprinzipien.**

Magische Attribute und Methoden in Python

Mehr erkunden :

https://www.tutorialsteacher.com/python/magic-methods-in-python

https://www.geeksforgeeks.org/dunder-magic-methods-python/

https://python-course.eu/oop/magic-methods.php

https://www.datacamp.com/tutorial/introducing-python-magic-methods