Klassen, Instanzen und Objekte

Einführung

In Python ermöglichen Klassen die Objektorientierte Programmierung (OOP). Klassen bieten eine Methode zur Bündelung von Daten und Funktionalitäten. Eine Klasse ist eine Blaupause für Objekte, und eine Instanz ist ein einzelnes, konkretes Objekt, das nach dieser Blaupause erstellt wurde.

Grundlegende Konzepte

Klassen definieren

Eine Klasse in Python wird mit dem Schlüsselwort `class` definiert, gefolgt von einem Klassennamen und einem Doppelpunkt. Innerhalb der Klasse definieren Sie Methoden (Funktionen), die das Verhalten der Klasse beschreiben.

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

    def gruessen(self):
        return f"Hallo, mein Name ist {self.name}!"
```

Instanzen erstellen

Um eine Instanz einer Klasse zu erstellen, rufen Sie einfach den Klassennamen auf, als ob es sich um eine Funktion handeln würde, und übergeben die notwendigen Argumente.

```
meine_instanz = MeineKlasse("Pythonista")
print(meine_instanz.gruessen())
```

Attribute und Methoden

- **Attribute** sind Daten, die einer Instanz zugeordnet sind. Im obigen Beispiel ist `name` ein Attribut von `MeineKlasse`.
- **Methoden** sind Funktionen, die das Verhalten der Instanzen einer Klasse definieren. `gruessen` ist eine Methode in `MeineKlasse`.

Erweiterte Konzepte

Vererbung

In Python kann eine Klasse von einer anderen erben, was bedeutet, dass sie Methoden und Attribute der Basisklasse wiederverwenden kann.

```
class SpezialisierteKlasse(MeineKlasse):
    def __init__(self, name, spezialitaet):
        super().__init__(name)
        self.spezialitaet = spezialitaet

    def spezialitaet_anzeigen(self):
        return f"Meine Spezialität ist {self.spezialitaet}."
```

Polymorphismus

Polymorphismus in Python ermöglicht es, dass Methoden in Kindklassen, die eine Methode mit demselben Namen wie die ihrer Basisklasse haben, eine andere Funktionalität bieten.

```
class AndereKlasse(MeineKlasse):
    def gruessen(self):
        return f"Guten Tag! Ich bin {self.name}."
```

Kapselung

Kapselung in Python wird oft durch Konvention erreicht, indem Attribut- oder Methodennamen mit einem Unterstrich (_) beginnen, um sie als privat zu markieren.

```
class KapselungBeispiel:
    def __init__(self):
        self.__privat_attribut = "Ich bin privat!"

    def __privat_methode(self):
        return "Dies ist eine private Methode!"

    def oeffentlich_methode(self):
        return f"Öffentliche Methode kann auf das {self.__privat_attribut}
zugreifen und die {self.__privat_methode()} aufrufen."
```

Zusammenfassung

In diesem Tutorial haben wir die Grundlagen von Klassen, Instanzen und Objekten in Python behandelt. Wir haben gesehen, wie man Klassen definiert, Instanzen erstellt und die Konzepte der Vererbung, des Polymorphismus und der Kapselung nutzt.