Abstrakte Klassen in Python

Definition: Eine abstrakte Klasse in Python ist eine Klasse, die nicht direkt instanziiert werden kann und zur Vererbung vorgesehen ist. Sie definiert eine Blueprint für andere Klassen.

Zweck: Sie bieten eine Möglichkeit, die Implementierung bestimmter Methoden in abgeleiteten Klassen (Child Class) zu erzwingen.

Vorteile abstrakter Klassen:

- **Methodenimplementierung erzwingen:** Stellt sicher, dass abgeleitete Klassen bestimmte Methoden implementieren.
- **Code-Wiederverwendbarkeit fördern:** Gemeinsame Schnittstellen für eine Gruppe verwandter Klassen.
- **Code-Organisation verbessern:** Verdeutlicht, welche Methoden überschrieben werden müssen.

Abstrakte Klassen in Python

Code-Beispiel:

```
from abc import ABC, abstractmethod

class Tier(ABC):
    @abstractmethod
    def laut(self):
        pass

class Hund(Tier):
    def laut(self):
        return "Wau"

class Katze(Tier):
    pass

c = Katze()
```

TypeError: Benötigen implementation für abstract method 'laut'

Modul: abc-Modul in der Python-Standardbibliothek. https://docs.python.org/3/library/abc.html
ABC : Klasse, die vom abc-Modul bereitgestellt werden, um abstrakte Basisklassen zu definieren.
@abstractmethod: Dekorator, um Methoden als abstrakt zu deklarieren.

Abstrakte Klassen in Python

Was sind Dekorateure?
Dekorateure sind eine sehr mächtige und ausdrucksstarke Möglichkeit in Python, das Verhalten von Funktionen oder Klassen zu modifizieren.

Ein Dekorateur nimmt eine Funktion als Argument und gibt eine neue Funktion zurück,
die in der Lage ist, zusätzliche Funktionalitäten vor und nach der Ausführung der ursprünglichen Funktion auszuführen. Das macht es möglich, Funktionen dynamisch zu erweitern.

```
def my_decorator(func):
    def wrapper():
        print("Vor dem Funktionsaufruf")
        func()
        print("Nach  dem Funktionsaufruf")
    return wrapper

@my_decorator
def say_hello():
    print("Hello!")

say_hello()
```

@my_decorator ist syntaktischer Zucker für say_hello = my_decorator(say_hello).

https://www.freecodecamp.org/news/python-decorators-explained/
https://www.geeksforgeeks.org/decorators-in-python/

Abstrakte Klassen - Praxisbeispiel Zahlungssystem

```
from abc import ABC, abstractmethod

class Zahlung(ABC):
    @abstractmethod
    def zahle(self, betrag):
        pass

    @abstractmethod
    def prüfe_zahlung(self):
        pass

class Kreditkartenzahlung(Zahlung):
    def zahle(self, betrag):
        print(f"Bearbeite Kreditkartenzahlung von {betrag}")

    def prüfe_zahlung(self):
        print("Überprüfe Kreditkartenzahlung")
```

Basisklasse: Die abstrakte Klasse, die die Vorlage bereitstellt.

Abgeleitete Klassen: Klassen, die die abstrakten Methoden implementieren.

Beispiel: Zahlung als Basisklasse mit Kreditkartenzahlung und anderen Zahlungstypen (weitere Kindklassen der Klasse Zahlung) als abgeleitete Klassen.

Abstrakte Klassen in Python

Wichtige Punkte

Instanziierung: Abstrakte Klassen können nicht instanziiert werden.

Implementierung: Alle abstrakten Methoden müssen in den abgeleiteten Klassen implementiert werden.

Flexibilität: Abstrakte Klassen bieten Flexibilität im Klassendesign und erzwingen konsistente Schnittstellen.

Mehr erkunden :

https://docs.python.org/3/library/abc.html

https://www.geeksforgeeks.org/abstract-classes-in-python/