Verwendung von `import` in Python

Einführung

Das `import`-Statement in Python wird verwendet, um Zugriff auf Code aus anderen Modulen oder Bibliotheken zu erhalten. Dies ermöglicht die Wiederverwendung von Code und trägt zur Modularität und Wartbarkeit von Programmen bei.

Grundlegender Import

Ein ganzes Modul importieren

```
import modulname
```

Beispiel

```
import math
print(math.sqrt(16))  # Ausgabe: 4.0
```

Importieren mit Alias

Manchmal ist es nützlich, einem importierten Modul einen anderen Namen (Alias) zu geben, um Schreibarbeit zu sparen oder Namenskonflikte zu vermeiden.

Syntax

```
import modulname as alias
```

Beispiel

```
import math as m
print(m.sqrt(16))  # Ausgabe: 4.0
```

Importieren spezifischer Funktionen

Es ist möglich, spezifische Funktionen, Klassen oder Variablen aus einem Modul zu importieren, anstatt das gesamte Modul zu laden.

Syntax

```
from modulname import funktion
```

Beispiel

```
from math import sqrt
print(sqrt(16))  # Ausgabe: 4.0
```

Importieren aller Inhalte eines Moduls

Mit dem `*`-Symbol können alle Funktionen und Variablen eines Moduls importiert werden. Dies sollte jedoch vermieden werden, da es zu Namenskonflikten führen kann.

Syntax

```
from modulname import *
```

Eigene Module erstellen und importieren

Ein Modul in Python ist einfach eine Datei mit der Endung `.py`. Man kann eigene Module erstellen, indem man Code in eine Datei schreibt und diese dann mit `import` in anderen Python-Dateien verwendet.

Beispiel

Angenommen, Sie haben eine Datei `meinmodul.py` mit folgendem Inhalt:

```
# meinmodul.py
def gruß():
    print("Hallo Welt!")
```

Sie können dieses Modul dann wie folgt importieren:

```
import meinmodul
meinmodul.gruß()  # Ausgabe: Hallo Welt!
```