das Import Statement

Pakete und Module importieren

Überblick

Mit dem import Statement lassen sich Module in den Code importieren und damit zugänglich machen.

Die import Statements sollten immer ganz oben in der Datei notiert werden. Es gibt allerdings sehr seltene Ausnahmen von dieser Regel.

import modulename

Der einfachste Zugriff auf ein Modul ist der komplette Import eines Moduls mit import modulename. Modulname ist ein eigener Namespace.

Wichtig ist hier zu verstehen, dass wir nun Zugriff auf die Methoden des Moduls über den Namespace modulname haben. Das ist von Vorteil, um im Code zu sehen, welche Methode oder Funktion aus welchem Modul kommt.

Wir import das Modul random und greifen über dessen Namespace auf die Funktion randint zu.

```
Import random
random.randint(3, 5)
```

from modulname import functionname

Eine weitere Möglichkeit besteht darin, eine Funktion direkt aus einem Modul zu importieren. Der Vorteil des Namespace geht damit allerdings verloren. Wir können nicht intuitiv sagen, woher die Funktion stammt. Zudem könnte die importierte Funktion mit einer Funktion gleichen Namens kollidieren.

Wir import die Funktion randint aus dem Modul random und greifen direkt auf sie zu.

```
from random import randint
randint(3, 5)
```

Prüfen, welche Module geladen sind

Python lädt per default immer einige Module selbständig. Um zu prüfen, welche Module aktuell schon geladen sind, kann man im sys.modules-Dictionary nachsehen.

```
import sys

for key, value in sys.modules.items():
    print(key, value)
```

from modulname import *

Alle Funktionen direkt aus einem Modul zu importieren, würde mit diesem Statement gehen. Das gilt allerdings als bad practice, da der Sternchenimport viele unserer eigenen Funktionen überschreiben könnte. Nur in Ausnahmefällen sollte man das tun.

Wir import alle Funktionen aus dem Modul random und greifen direkt auf sie zu.

```
from random import *
randint(3, 5)
randrange(3, 5)
```

From modulanme import func as alias

Beim import können wir der importierten Funktion ein Alias geben. Es gibt selten einen Grund, einer Funktion einen Alias zu geben. Ein Anwendungsfall wäre, wenn die Funktion im globalen Scope schon exisitert.

```
from random import randint as randy

randy(3, 5)
```

import modulename as alias

Beim import eines Moduls können wir auch dem Modul selbst ein Alias geben. Das ist gängiger Anwendungsfall beim Importieren von Modulen und Paketen aus der Standard-Bibliothek bzw. von Paketen Drittanbierter.

Wir importieren das Paket numpy und geben dem Paket im Code den Alias np. Damit sind Methoden und Module über diesen Namespace erreichbar.

```
import numpy as np
np.random.uniform(3, 20)
```

```
import itertools as it
it.product(3, 7)
```