Module und Imports in Python

Einführung und Anwendung

Ziele des Tages

- Verstehen der verschiedenen Arten von Imports in Python
- Anwendung der Imports in unterschiedlichen Situationen
- Erstellung eigener Module und Pakete

Einführung in Module und Imports

- **Funktionen**: Wiederverwendbare Codeblöcke, die eine bestimmte Aufgabe erfüllen.
- **Module**: Einzelne Python-Dateien, die z.B. Funktionen und Variablen enthalten. (z.B. `mymodule.py`)
- **Pakete**: Verzeichnisse, die Module und eine `__init__.py` Datei enthalten

---

Was ist der Unterschied zwischen Modulen und Paketen?

Standardbibliothek vs. Drittanbieter-Module

Standardbibliothek: Eine Sammlung von Modulen und Paketen, die mit der Python-Installation mitgeliefert werden. Diese decken viele grundlegende Aufgaben ab (z.B. math, os, sys).

Beispiel:

```
import os, import sys
```

Drittanbieter-Module: Module und Pakete, die nicht Teil der Standardbibliothek sind und von der Python-Community entwickelt wurden. Sie können über den Paketmanager pip installiert werden.
Beispiel:

```
import numpy, import requests
```

Wie ist die Installation von Drittanbieter-Modulen?

Die verschiedenen Arten von Imports in Python

Beispiel: Einfache Moduldatei

```python
# mymodule.py
def greet():
    return "Hello, world!"
```

Allgemeine Imports

- Syntax: `import modulename`
- Beispiel:

```
import mymodule
print(mymodule.greet())
```

Aufgabe:

Erstelle ein Modul math_utils.py, das eine Funktion add(a, b) und eine Funktion multiply(a, b) enthält. Importiere das gesamte Modul in main.py und nutze beide Funktionen, um zwei Zahlen zu addieren und zu multiplizieren.

Selektive Imports

- Syntax: `from modulename import name`
- Beispiel:

```
from mymodule import greet
print(greet())
```

Alias-Imports

- Syntax: `import modulename as alias`
- Beispiel:

```
import mymodule as mm
print(mm.greet())
```

Kombinierte Imports

- Syntax: `from modulename import name as alias`
- Beispiel:

```
from mymodule import greet as gr
print(gr())
```

Die Datei `__init__.py` und eigene Module

- Zweck: Kennzeichnung von Verzeichnissen als Python-Pakete
- Initialisierungslogik für Pakete

Beispiel: Paketstruktur

mypackage/
init.py

module1.py
module2.py

```
# __init__.py
from .module1 import func1
from .module2 import func2
```

Aufgabe:

Erstelle ein Paket my_math_package, das zwei Module basic_operations.py und advanced_operations.py enthält.

- Im basic_operations.py Modul gibt es Funktionen add(a, b) und subtract(a, b),
- im advanced_operations.py Modul gibt es Funktionen power(base, exp) und sqrt(value).
- Definiere in der **init**.py des Pakets, dass alle diese Funktionen importiert werden. Nutze das Paket in main.py.

Quizfragen

1. Wie importiert man ein gesamtes Modul?
2. Wie importiert man eine spezifische Funktion aus einem Modul?
3. Was ist der Vorteil eines Alias-Imports?
4. Was ist die Rolle der Datei `__init__.py` in einem Paket?