# Lektion 13 – Module selbst erstellen

## Nachschlagen bei…

W3Schools: Modules

## Anweisungen

Studiere das genannte Kapitel in W3Schools und die folgenden Erläuterungen. Beantworte die Fragen in diesem Dokument und kopiere den Code der Aufgaben hierher.

## Erläuterungen

Ein umfangreiches Programm kann viele Funktionen, Konstanten, Listen und Dictionaries haben und dadurch leicht unübersichtlich werden. Lösung: man kopiert die Definitionen der Funktionen, Listen, etc. in eine eigene .py-Datei (zB meinmodul.py) und importiert diese in das eigentliche Programm (zB import meinmodul). Die Verwendung der Funktionen geht genau so wie in der letzten Lektion. Das lohnt sich vor allem dann, wenn man das Modul später auch für andere Programme verwenden kann.

### Beispiel - Modul „rechnen“

def addieren(a,b):
 c=a+b
 print(a, "+", b, "=",c)
Die schon bekannte Funktion „addieren()“ wird in einer Datei „rechnen.py“ gespeichert.

Der folgende Code wird in einer anderen Datei (zB rechenaufgaben.py) gespeichert:

import rechnen as r
r.addieren(6,4)
r.addieren(3,5) *Probier aus, was dieser Code macht*

## Fragen

Mit welcher Syntax wird die Funktion „meinefunktion()“ aus dem importierten Modul „meinmodul“ aufgerufen?

Mit welcher Syntax muss das Modul „meinmodul“ eingebunden werden, damit man die enthaltenen Funktionen mit dem alias „m“ (zB m.meinefunktion()) aufgerufen werden kann?

Mit welcher Anweisung kann man auflisten, welche Funktionen und Variablen in einem Modul enthalten sind?

## Aufgabe

Erweitere das Modul „rechnen“ aus dem Beispiel um subtrahieren() und multiplizieren(). Ändere das Programm rechenaufgaben.py so ab, dass alle drei Rechenarten vorkommen. Die Ausgabe soll dann so aussehen:

6 + 4 = 10
6 - 4 = 2
6 \* 4 = 24

## Fleißaufgabe

Erzeuge ein Modul „vektorrechnung“, das die Funktion vadd(a,b) mit Rückgabewert aus dem Beispiel 11 enthält. Erweitere das Modul um Vektorsubtraktion (vsub(a,b)), Skalarmultiplikation (vmult(a,b)) und Multiplikation mit einer Zahl (smult(a,z)). Erstelle eine Datei „vektoraufgaben.py“ in der das Modul „vektorrechnung“ eingebunden wird und der Reihe nach die Funktionen aufgerufen werden. Die Ausgabe soll dann so aussehen:

[4, 3, 1, 1] + [2, 6, 5, 1] = [6, 9, 6, 2]
[4, 3, 1, 1] - [2, 6, 5, 1] = [2, -3, -4, 0]
[4, 3, 1, 1] \* [2, 6, 5, 1] = 32
[4, 3, 1, 1] \* 5 = [20, 15, 5, 5]