

Praktikum 4: Funktionen

Hinweis:

- Nutzen Sie Kommentare um Ihren Code verständlich zu machen.
- Der Debugger kann Ihnen dabei helfen den Programmablauf besser zu verstehen

Ziel des heutigen Praktikums ist es, Ihnen den Umgang mit Funktionen und der Übergabe von Parametern zu verdeutlichen. Starten Sie VS-Code wie gewohnt. Speichern Sie jede Aufgabe in einer eigenen Pythondatei ab.

Aufgabe 1

Die erste Aufgabe besteht darin, unterschiedliche Möglichkeiten kennenzulernen, wie eine Funktion zur Berechnung der Fakultät $n!$ in Python programmiert werden kann und welche Vor- und Nachteile damit verbunden sind.

a) Eine mögliche mathematische Definition der Fakultät ist

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n = \prod_{k=1}^n k$$

Definieren Sie bitte eine Python-Funktion

`fakultaet_iterativ(n)`

die eine Schleife Ihrer Wahl benutzt, um das Ergebnis nach obiger Vorschrift zu berechnen und zurückzugeben.

Geben Sie 4! Und 10.000! aus.

b) Die Fakultätsfunktion kann aber auch rekursiv definiert werden, indem das Ergebnis für einen Startwert festgelegt wird und alle weiteren Ergebniswerte auf diesen Startwert zurückgeführt werden.

$$n! = \begin{cases} 1 & \text{falls } n = 0 \\ n \cdot (n-1)! & \text{sonst} \end{cases}$$

Definieren Sie bitte eine Python-Funktion

`fakultaet_rekursiv(n)`

die diese Berechnungsvorschrift nachbildet. Implementieren Sie dazu zunächst eine Verzweigung, die für den Fall $n=0$ das vorgegebene Ergebnis liefert. Andernfalls soll sich die Funktion selbst mit einem angepassten Parameterwert aufrufen und damit das richtige Ergebnis berechnen.

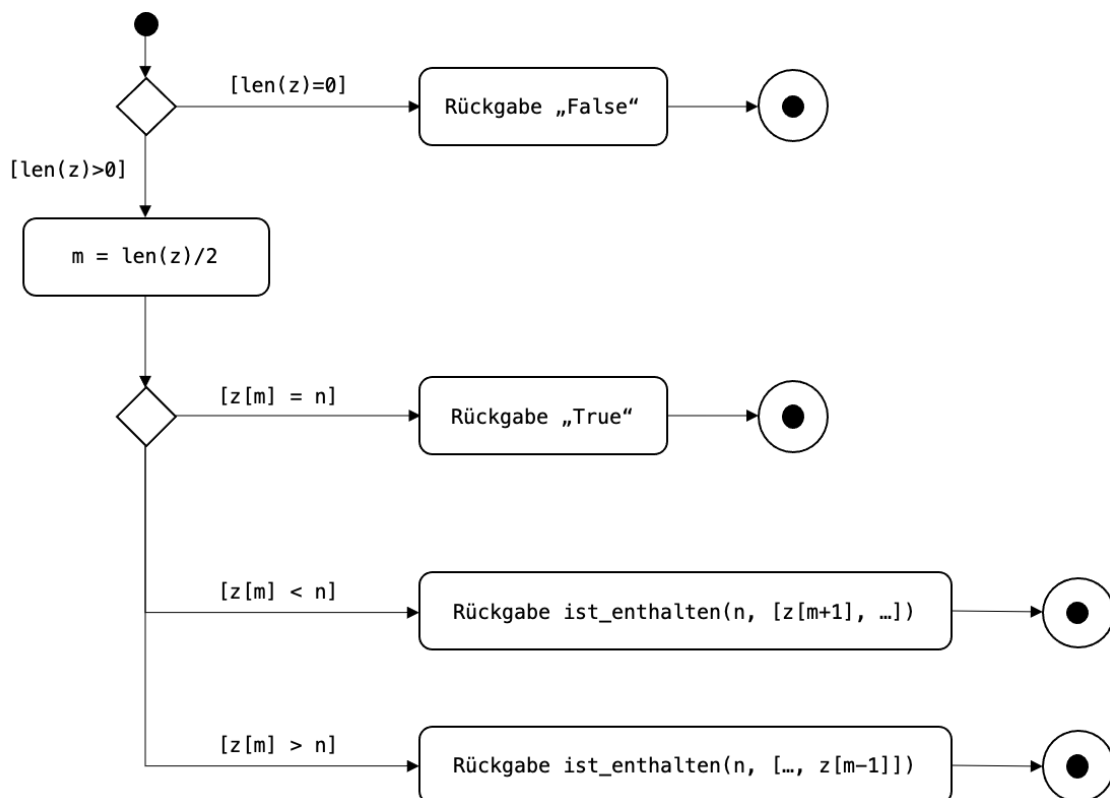
Geben Sie auch mit Hilfe dieser Funktion 4! Und 10.000! aus. Was beobachten Sie? Wie erklären Sie sich das Verhalten?

Aufgabe 2

- a) Gegeben ist eine aufsteigend sortierte Liste ganzer Zahlen z . Ihre Aufgabe ist es, eine Funktion

$\text{ist_enthalten}(n, z)$

zu implementieren, die True zurückgibt, wenn n in z enthalten ist, andernfalls False. Dabei soll aber die Anzahl der Zugriffe minimiert werden, indem der folgende rekursive Algorithmus verwendet wird:



Überprüfen Sie mit Ihrer Funktion, ob in der Liste

$z = [1, 2, 3, 5, 8, 13, 21, 34]$

die Werte 7 und 5 enthalten sind.

Wie viele Zugriffe sind maximal notwendig, im festzustellen, ob ein Element in der Liste ist?

- b) Schreiben Sie die Funktion so um, dass die gleiche Lösungsidee mit einer Schleife (iterativ) verfolgt wird, aber kein rekursiver Aufruf nötig ist. Verwenden Sie nicht den „in-Operator“.