5.6 Rekursive Funktionen

Donnerstag, 20. April 2023 16:24

5.6. Rekursive Funktionen

If you don't understand recursion read this sentence again!

Wie bereits erwähnt, darf eine Funktion eine andere aufrufen. Es ist sogar erlaubt, dass sich eine Funktion selbst wieder aufruft. Dies kann für einige Dinge nützlich sein, wie zum Beispiel hier:

```
def countdown(n):
    if n <=0:
        print("Bumm!")
    else:
        print(n)
        countdown(n-1)
```

Wenn n gleich 0 oder negativ ist, wird das Wort „Bumm" ausgegeben. Ansonsten wird n ausgegeben, eine Funktion mit dem Namen countdown wird aufgerufen (dieselbe Funktion!) und n-1 wird als Argument übergeben.
Beispiel: countdown(2)
Da n größer ist als 0, gibt die Funktion 2 aus und ruft sich selbst auf...
Die Ausführung von countdown beginnt mit n=1. Da n größer ist als 0, gibt die Funktion 1 aus und ruft sich selbst auf....
Die Ausführung beginnt mit n=0. Da n nicht größer als 0 ist, gibt die Funktion „Bumm!" aus und kehrt zurück.
Die Ausgabe sieht also so aus:
3
2
1
Bumm!

Eine Funktion, die sich selbst aufruft, nennt man **rekursiv** und den Vorgang nennt man **Rekursion**.

Rekursive Algorithmen eignen sich besonders gut, um hierarchische Datenstrukturen zu verarbeiten, also z.B. um alle Verzeichnisse der Festplatte oder Baumstrukturen zu durchsuchen.
Schauen wir uns doch einmal ein etwas kompliziertes Beispiel einer Rekursion an: Die Fibonacchi - Zahlen:

Dabei ist diese Fibonacci-Folge simpel: Der Beginn ist bei null und eins, danach ist jede Zahl die Summe der beiden unmittelbar vorangegangenen Zahlen. Also: 0+1=1; 1+1=2; 1+2=3; 2+3=5; 3+5=8 und so weiter.

Ganz allgemein können wir festhalten: **fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)**

Implementieren Sie eine rekursive Funktion, die die Fibonacci Folge wiedergibt

Kontrolle:

n	f_n	n	f_n	n	f_n	n	f_n	n	f_n
1	1	11	89	21	10 946	31	1 346 269	41	165 580 141
2	1	12	144	22	17 711	32	2 178 309	42	267 914 296
3	2	13	233	23	28 657	33	3 524 578	43	433 494 437
4	3	14	377	24	46 368	34	5 702 887	44	701 408 733
5	5	15	610	25	75 025	35	9 227 465	45	1 134 903 170
6	8	16	987	26	121 393	36	14 930 352	46	1 836 311 903
7	13	17	1 597	27	196 418	37	24 157 817	47	2 971 215 073
8	21	18	2 584	28	317 811	38	39 088 169	48	4 807 526 976
9	34	19	4 181	29	514 229	39	63 245 986	49	7 778 742 049
10	55	20	6 765	30	832 040	40	102 334 155	50	12 586 269 025

```python
def fibonacci(fibonacci_numbers:list = [0,1], counter = 50):
    if counter > 0:
        counter -= 1
        fibonacci_numbers.append(fibonacci_numbers[-1]+fibonacci_numbers[-2])
        fibonacci_numbers, counter = fibonacci(fibonacci_numbers=fibonacci_numbers, counter=counter)

    return fibonacci_numbers, counter
print(fibonacci()[0])
```