

Prof. Dr. Claudia Müller-Birn, Barry Linnert

Objektorientierte Programmierung, SoSe 17

Übung 04

TutorIn: Thierry Meurers
Tutorium 10

Stefaan Hessmann, Jaap Pedersen, Mark Niehues

22. Mai 2017

1 Rekursion in Python

Listing 1: Code zu Aufgabe 1

```
1 """
2 Aufgabe 1
3 """
4
5
6 def rekursion(counter=1, error_occured = False):
7     """
8     Calls itself until an error occurs.
9     Prints the number of recursion-steps.
10
11     Parameters
12     -----
13     counter : int
14         should not be set
15
16     error_occured : boolean
17     Determines if an error already occurred
18
19     Returns
20     -----
21     None
22     """
23     counter += 1
24     if not error_occured:
25         try:
26             rekursion(counter)
27         except:
28             if not error_occured:
29                 print("Nach {} Rekursionen ist die Rekursionstiefe erreicht".format(counter))
30                 error_occured = True
31
32
33 if __name__ == '__main__':
34     rekursion()
```

2 Türme von Hanoi in Python

Listing 2: Code zu Aufgabe 2

```
1 """
2 Aufgabe 2
3 Hessmann, Niehues, Pedersen
4
5 Demonstrates the Towers of Hanoi using a recursive algorithm.
6
7 Similar to solution:
8 http://www.python-kurs.eu/tuerme_von_hanoi.php
9 """
10
11 def hanoi(n, source, helper, target, rekursionstiefe = 0):
12     if n > 0:
13         # move tower of size n - 1 to helper:
14         hanoi(n - 1, source, target, helper, rekursionstiefe + 1)
15
16         # move disk from source peg to target peg
17         if source[0]:
18             print("Moving {} from {} to {}".format(source[0][-1], source[1], target[1]))
19             target[0].append(source[0].pop())
20
21         # Parameters need to be seperated for printing
22         afg = ""
23         hlp = ""
24         zl = ""
25         for i in [source, target, helper]:
26             if i[1] == "Anfang":
27                 afg = i[0]
28             elif i[1] == "Ziel":
29                 zl = i[0]
30             elif i[1] == "Hilfsstab":
31                 hlp = i[0]
32             print("Source: {} \t Helper: {} \t Target: {}".format(afg, hlp, zl))
33
34         # move tower of size n-1 from helper to target
35         hanoi(n - 1, helper, source, target, rekursionstiefe + 1)
36
37 n = int(input("How many discs?:\n"))
38
39 source = list(range(1,n+1)) # Creates list from 1 to n
40 source = source[::-1], "Anfang" # Invertes the order
41 target = ([], "Ziel")
42 helper = ([], "Hilfsstab")
43 hanoi(n,source,helper,target)
```

3 Auswirkung der Rekursionstiefe in Python

Rein theoretisch wird die Berechenbarkeit aus Aufgabe 2 auch durch die Rekursionstiefe begrenzt. Da die maximale Rekursionstiefe allerdings n ist wird die Berechenbarkeit in der Praxis allein vom hohen Zeitaufwand begrenzt. Laut Aufgabe 1 war auf diesem PC eine Rekursionstiefe von 1000 möglich. Bei 30 Scheiben dauert die Berechnung allerdings schon 34 Jahre laut https://de.wikipedia.org/wiki/Türme_von_Hanoi#Praktische_Unl.C3.B6sbarkeit