Freie Universität Berlin

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

```
"""
Aufgabe 1
"""


def rekursion(counter=1, error_occured = False):
    """
    Calls itself until an error occurs.
    Prints the number of recursion-steps.

    Parameters
    ----------
    counter : int
        should not be set

    error_occured : boolean
  Determines if an error already occured

    Returns
    -------
    None
    """
    counter += 1
    if not error_occured:
        try:
            rekursion(counter)
        except:
            if not error_occured:
             print("Nach {} Rekursionen ist die Rekursionstiefe erreicht".format(counter))
             error_occured = True


if __name__ == '__main__':
    rekursion()
```

2 Türme von Hanoi in Python

Listing 2: Code zu Aufgabe 2

```
"""
Aufgabe 2
Hessmann, Niehues, Pedersen

Demonstrates the Towers of Hanoi using a recursive algorithm.

Similar to solution:
http://www.python-kurs.eu/tuerme_von_hanoi.php
"""

def hanoi(n, source, helper, target, rekursionstiefe = 0):
    if n > 0:
        # move tower of size n - 1 to helper:
        hanoi(n - 1, source, target, helper, rekursionstiefe = rekursionstiefe + 1)

        # move disk from source peg to target peg
        if source[0]:
            print("Moving {} from {} to {}".format(source[0][-1], source[1], target[1]))
            target[0].append(source[0].pop())

            # Parameters need to be seperated for printing
            afg = ""
            hlp = ""
            zl = ""
            for i in [source, target, helper]:
                if i[1] == "Anfang":
                    afg = i
                elif i[1] == "Ziel":
                    zl = i
                elif i[1] == "Hilfsstab":
                    hlp = i
            print("{}: {}\t {}: {}\t {}: {}".format(afg[1], afg[0], hlp[1], hlp[0], zl[1],
    zl[0]))

        # move tower of size n-1 from helper to target
        hanoi(n - 1, helper, source, target, rekursionstiefe = rekursionstiefe + 1)


if __name__ == '__main__':
    n = int(input("How many discs?:\n"))

    source = list(range(1,n+1)) # Creates list from 1 to n
    source = (source[::-1], "Anfang")       # Invertes the order
    target = ([], "Ziel")
    helper = ([], "Hilfsstab")
    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