Rekursive Funktionen (1)

- In der Mathematik gibt es rekursive Definitionen z.B. von Folgen. Beispiele:

  $a_1 = 1, \quad a_k = a_{k-1} \cdot q$ für $k = 2, 3, \ldots$ (geometrische Folge: $1, q, q^2, q^3, \ldots$)

  $a_1 = 1, \quad a_k = a_{k-1} + 1/2^{k-1}$ für $k = 2, 3, \ldots$ (Reihe, die gegen 2 konvergiert)

- In Programmiersprachen kann dies mit **rekursiven Funktionen** nachgebildet werden. Beispiele:

```
def geo(k, q):
    if k == 1:
        return 1
    else:
        return geo(k - 1, q) * q
```

```
def row(k):
    if k == 1:
        return 1
    else:
        return row(k - 1) + 1 / (2 ** (k-1))
```

- Eine rekursive Funktion ruft sich also selbst auf.

Münchhausen, Illustration von Theodor Hosemann (1807-1875), Bildquelle: Wikipedia (gemeinfrei), https://upload.wikimedia.org/wikipedia/de/6/64/M%C3%BCnchhausen-Sumpf-Hosemann.PNG

Rekursive Funktionen (2)

- Weiteres Beispiel:

```
def sum(fro, to):
    if fro == to:
        return fro
    else:
        return fro + sum(fro + 1, to)
```

- Beachten Sie, dass die Aufrufe von sum(3, 7), sum(4, 7), sum(5, 7), sum(6, 7) erst abgeschlossen werden, nachdem die Rekursion mit dem Aufruf sum(7, 7) abbricht.

Aufruftabelle für sum(3, 7):

Aufruf	Rückgabewert
sum(3, 7)	3 + sum(4, 7) = 25
sum(4, 7)	4 + sum(5, 7) = 22
sum(5, 7)	5 + sum(6, 7) = 18
sum(6, 7)	6 + sum(7, 7) = 13
sum(7, 7)	7

Rekursive Funktionen (3)

- Was passiert, wenn der erste Fall vergessen wird?

```
def sum(fro, to):
    return fro + sum(fro + 1, to)
```

Die Rekursion terminiert nicht!

- Deshalb:
  - Nicht den **Abbruchfall** vergessen!
  - Vergewissern Sie sich, dass der Abbruchfall die Rekursion für alle möglichen Eingaben beendet!

Aufruf
sum(3, 7)
sum(4, 7)
sum(5, 7)
sum(6, 7)
sum(7, 7)
sum(8, 7)
sum(9, 7)

Rekursive Funktionen (4)

- Der Vorteil rekursiver Algorithmen ist, dass sie keine Schleifen benötigen und deshalb oft sehr einfach zu erstellen sind.
- Beginnen Sie bei der Erstellung rekursiver Algorithmen mit der Rekursionsformel. Beispiele:

  fac(1) = 1, fac(k) = k * fac(k – 1)

  fib(0) = 0, fib(1) = 1, fib(k) = fib(k – 1) + fib(k – 2)

  Der Abbruchfall tritt in den Beispielen ein, wenn k gleich 1 bzw. gleich 0 ist.
- In Fällen, in denen der Parameter eine Zeichenkette ist, könnte der Abbruchfall eintreten, z.B. wenn die Zeichenkette die leere Zeichenkette ist. Beispiel:

```
def length(s):
    if s == "":
        return 0
    else:
        return 1 + length(s[1:])
```

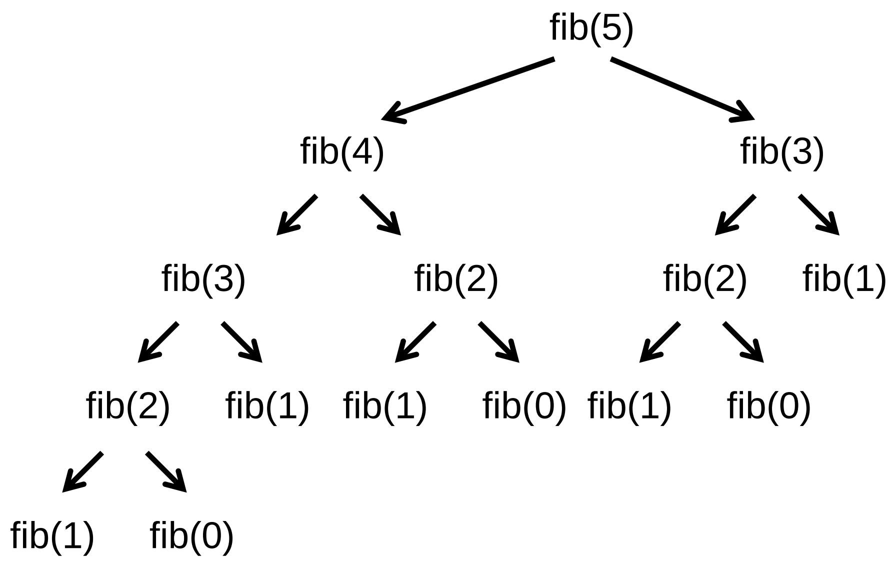
Rekursive Funktionen (5)

- Ein Nachteil ist, dass manche rekursiven Algorithmen komplexer in der Ausführung sind und daher mehr Laufzeit in Anspruch nehmen.
- Beispiel:

```
def fib(k):
    if k == 0:
        return 0
    elif k == 1:
        return 1
    else:
        return fib(k - 1) + fib(k - 2)
```

  - fib(3) wird zweimal berechnet,
    fib(2) dreimal, fib(1) fünfmal, fib(0) dreimal
- 🕮 Kapitel 15 in (Klein 2018)

Aufrufbaum für fib(5):



Rekursive Funktionen (6)

Aufgaben

1. Erstellen Sie eine rekursive Funktion zur Berechnung des k-ten Gliedes der Folge 1, 2, 4, 7, 11, 16, 22, … Verwenden Sie die Rekursionsformel zafo(1) = 1, zafo(k) = k – 1 + zafo(k – 1).

2. Erstellen Sie eine rekursive Funktion zur Berechnung von ggt(a, b). Es sei vorausgesetzt, dass a und b positiv sind.

   Der Abbruchfall tritt ein, wenn a gleich b ist. Andernfalls wird ggt(a – b, b) oder ggt(b, b – a) aufgerufen, je nachdem, ob a größer als b ist oder nicht.

3. Erstellen Sie eine rekursive Funktion zur Prüfung, ob eine Zeichenkette word ein Palindrom ist.

   Der Abbruchfall tritt ein, wenn die Länge der Zeichenkette kleiner oder gleich 1 ist. Es gibt einen zweiten Abbruchfall, wenn das erste und das letzte Zeichen unterschiedlich sind. Andernfalls wird die Funktion mit dem Parameter word[1:len(word)-1] aufgerufen.

   Erstellen Sie eine Aufruftabelle für is_palindrom("annna").

Beispiel Brüche

- Bevor wir diesen Teil beenden, wollen wir ein komplettes Programm erstellen. Es soll Brüche addieren und multiplizieren können.
- Der Benutzer gibt die gewünschte Rechnung ein, z.B. -5/4 + 7/6, und das Programm antwortet mit dem Ergebnis, im Beispiel = -1/12. Das Ergebnis soll gekürzt sein.
- In einem Skript bruch.py implementieren wir Funktionen für die Addition, die Multiplikation und das Kürzen. Für das Kürzen brauchen wir die Berechnung des ggt.
- Da die Eingabe eine Zeichenkette ist, muss sie zuerst in ihre Bestandteile zerlegt werden. Das und den Aufruf der Addition bzw. der Multiplikation implementieren wir im Hauptprogramm bruch_app.py
- Wir erstellen das Programm in der Vorlesung. Die beiden Skripte werden danach auch auf Moodle gestellt.