Programmiervorkurs 2025

Rekursion

Vorkurs-Schnupsis

2. Oktober 2025

TU Darmstadt

Funktionen und Schleifen

Wiederholung

```
1  def zaehlen(name, buchstabe):
2    zaehler = 0
3    for zeichen in name:
4      if zeichen.lower() == buchstabe:
5        zaehler += 1
6    return zaehler
7
8  name = input('Name: ')
9  zahl = zaehlen(name, 'a')
10 print(f"Du hast {zahl} A's im Namen")
```

Funktionen und Schleifen

Wiederholung

- Kann man damit schon alles machen?

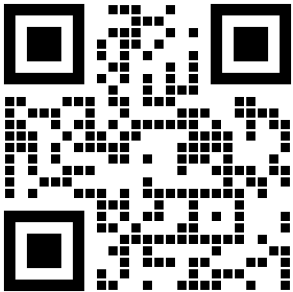
  $\Rightarrow$ Ja, aber es wird nicht schön...
- Tatsächlich können Funktionen noch einiges mehr
- Eine sehr wichtiges Eigenschaft wollen wir euch nicht vorenthalten:

  $\Rightarrow$ Funktionen können sich selber aufrufen!

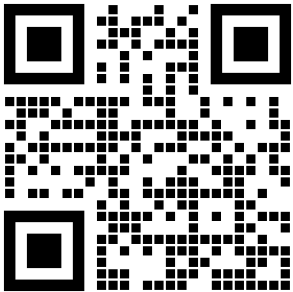
Evaluation

Evaluation

Bewerte unsere Vorlesung und Übung



d120.de/vkevalvl

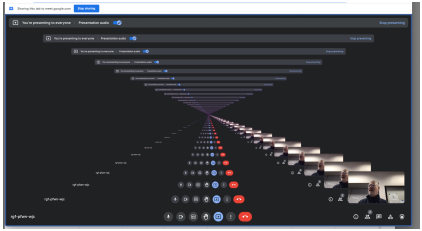


d120.de/vkevalue

Rekursion

Rekursion

Was ist das eigentlich?

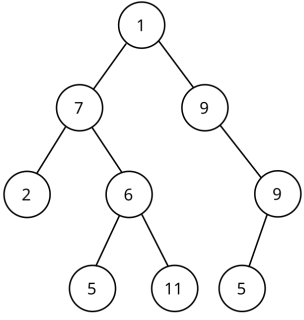


Rekursion im Alltag

Rekursion

Bruder, was das?

Rekursion in der Natur



Binärbäume in der Informatik

Rekursion

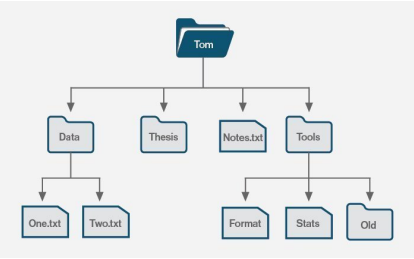
Was ist das eigentlich?

- Viele Dinge (insb. in der Mathematik oder Informatik) sind oft im Bezug auf sich selbst definiert
- Das nennt man eine *rekursive Definition*
- Beispiele dafür sind unter anderem:
  - Die Fibonacci-Folge
  - Die Fakultät
  - Der Programmcode, den wir schreiben
  - Dateisysteme

Rekursion

Was ist das eigentlich?

Ordner sind **rekursiv** definiert, d.h. diese können Unterordner haben



Rekursion in Dateisystemen

Definition

Wie definieren wir Rekursion?

Die wesentlichen Bestandteile:

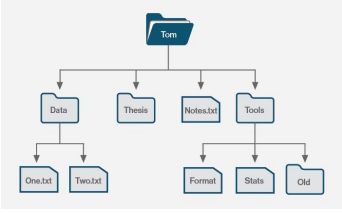
- Anfangs- bzw. Ausgangszustand,
  oft *Rekursionsanker* genannt
- Anweisungsabfolge, um von einem Zustand zum nächsten zu kommen,
  oft *Rekursionsschritt* genannt

Definition

Wie definieren wir Rekursion?

Beispiel Dateisystem:

- Szenario: Man möchte vom Wurzelknoten aus eine Datei suchen
- Dafür werden in jedem Ordner die Liste der Unterelemente durchlaufen und geprüft, ob die gesuchte Datei dabei ist
- Wurde die Datei gefunden oder es ist nicht möglich weiter abzusteigen, kehrt die Suchfunktion zur Ebene darüber zurück
- Überlegt euch in den davor gezeigten Beispielen, was Anker- und was Rekursionsschritt ist



Rekursion in Python

Wie definieren wir Rekursion in Python?

Die Fakultät ist rekursiv definiert als

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

In Python:

```python
def factorial(n):
  if n == 0:   # Rekursionsanker
    return 1

  # Rekursiver Funktionsaufruf von factorial
  return n * factorial(n-1)
```

Rekursion in Python

```python
def is_even(n):
  if n == 0: # Rekursionsanker
    return True
  return is_odd(n-1) # Rekursiver Funktionsaufruf

def is_odd(n):
  if n == 0: # Rekursionsanker
    return False
  return is_even(n-1) # Rekursiver Funktionsaufruf
```

Merke: Rekursion kann auch hin und her springen

Rekursion in Python

Die Fibonacci Folge ist rekursiv definiert als

$$fib(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

In Python:

```
1 def fibonacci(n):
2   if n <= 1:   # Rekursionsanker
3     return n
4   #      v -------------- v ------------ Rekursive Funktionsaufrufe
5   return fibonacci(n-1) + fibonacci(n-2)
```

Merke: Funktionen können sich auch **mehrmals** rekursiv aufrufen

Live-Coding

Rekursion in Python

Probleme

- Grundsätzlich lässt sich jede Schleife in Python auch mit Rekursion umsetzen
- Ausnahme: Endlosschleife bzw. Schleifen mit zu vielen Iterationen
- Python hat eine maximale Tiefe an Funktionsaufrufen von ungefähr 1000

Rekursion in Python

Probleme

Unendliche Rekursion

```
>>> def fun(num):
    fun(num+1)
>>> fun(0)
```

Was ist das Ergebnis?

Rekursion in Python

Probleme

Unendliche Rekursion

```
Traceback (most recent call last):
  File "<python-input-1>", line 1, in <module>
    fun(0)
    ~~~^^^
  File "<python-input-0>", line 2, in fun
    fun(num+1)
    ~~~^^^^^^^
  [Previous line repeated 988 more times]
RecursionError: maximum recursion depth exceeded
```

Definition

Probleme

Live-Coding

Rekursion

Quiz

- Wie viele Rekursionsanker braucht man?
- Wie macht man eine Funktion in Python rekursiv?
- Ist im Allgemeinen Rekursion oder Iteration sinnvoller?

Java

Ein Ausblick I

Java ist die nächste Programmiersprache, die ihr im Studium lernen werdet. Aber wie unterscheidet sich Java von Python?

- Statische Typisierung
  - Typen von Variablen, Rückgabetypen und Funktionsparametern müssen im Quelltext stehen
  - Zur Laufzeit kann dadurch weniger schief gehen
    $\Rightarrow$ Eure Programme sind robuster

Ein Ausblick II

- Objektorientierung
  - Funktionen auf Objekten: **Methoden**
  - Diese kennt ihr bereits, z.B. `"Hallo".upper()`
- Einrückung ist in Java irrelevant
  - Stattdessen nutzt man geschweifte Klammern und Semikola, um Code zu trennen
- Java wird kompiliert
  - Man führt nicht direkt den Quelltext aus, sondern Bytecode
  - Vor der Ausführung muss erst der Compiler aufgerufen werden (`javac`)

Variablen

Python:

```
1 x = 5
2 s = 'Lorem'
```

Java:

```
1 int x = 5;
2 String s = "Lorem";
```

Wichtiger Unterschied:
In Java können Strings nur mit `""` definiert werden, nicht mit `''`

Verzweigungen

Python:

```python
if x < 0:
    print("Negativ")
elif x > 0:
    print("Positiv")
else:
    print("Null")
```

Java:

```java
if (x < 0) {
    System.out.println("Negativ");
} else if (x > 0) {
    System.out.println("Positiv");
} else {
    System.out.println("Null");
}
```

Schleifen

while

Python:

```python
while n > 0:
    n = n // 2 - 1
```

Java:

```java
while (n > 0) {
    n = n / 2 - 1;
}
```

Zusätzlich gibt es eine do-while Schleife, diese lernt ihr dann im ersten Semester.

Schleifen

for-each

Python:

```python
fruits = ["apple", "orage", "banana"]
for f in fruits:
    print(f + "tasted good")
```

Java:

```java
String[] fruits = {"apple", "orange", "banana"};
for (String f : fruits) {
    System.out.println(f + "tasted good");
}
```

Schleifen

for mit Zähler

Python:

```python
for i in range(1, 11):
    print(str(i) + " squared is " + str(i*i))
```

Java:

```java
for (int i = 1; i < 11; ++i) {
    System.out.println(i + " squared is " + i*i)
}
```

Funktionen

Python:

```python
def say_hello(name):
    print("Hello " + name + "!")
    print("How are you?")
```

Java:

```java
void sayHello(String name) {
    System.out.println("Hello " + name + "!");
    System.out.println("How are you?");
}
```

Sprachen Idiomas Langues

Es gibt noch andere Sprachen

- Es gibt auch noch andere Sprachen und andere Paradigmen als das was wir kennengelernt haben
- Funktionale sprachen haben als basis funktionen und konstanten
- Stack Sprachen, basieren auf dem Prinzip eines Stacks
- Array Sprachen, basieren auf listen als nativen typ

Ende

Vielen Dank

Viel Spaß in der Orientierungswoche und im weiteren Studium!

Nachher ist noch eine Übung, ansonsten genießt das lange Wochenende.

Denkt an die Challenge, wir freuen uns über **jede** Abgabe.