Übungen zu Computergrundlagen WS 2014/2015

Übungsblatt 6: Python III

19.11.2014

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist
  - **Freitag, 28.11.2014, 10:00**
- Schickt die Lösungen bitte per Email an Euren Tutor.

Aufgabe 6.1: Funktionen, Rekursionen und Schleifen (10 Punkte)

- **6.1.1** Die Sequenz der Fibonacci-Zahlen ist wie folgt definiert:

$$\text{fib}(n) = \begin{cases} 0 & \text{falls } n = 0 \\ 1 & \text{falls } n = 1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{sonst} \end{cases} \quad (1)$$

  Schreibt eine Python-Funktion `fib1(n)`, die die n-te Fibonacci-Zahl wie in Gleichung 1 rekursiv berechnet. (2 Punkte)

- **6.1.2** Schreibt eine Funktion `fib2`, die die Fibonacci-Zahlen mit Hilfe einer Schleife berechnet, also ohne den rekursiven Aufruf der Funktion. (2 Punkte)

  Hinweis: Berechnet zunächst auf dem Papier von Hand die ersten paar Fibonacci-Zahlen. Das macht es einfacher, zu verstehen, wie man die Schleife implementieren kann.

- **6.1.3** Eine andere, aber äquivalente, Definition der Fibonacci-Zahlen sieht so aus:

$$\text{fib}_{a,b}(n) = \begin{cases} a & \text{falls } n = 0 \\ b & \text{falls } n = 1 \\ \text{fib}_{b,(a+b)}(n-1) & \text{sonst} \end{cases} \quad (2)$$

$$\text{fib}(n) = \text{fib}_{0,1}(n) \quad (3)$$

  Schreibt eine Python-Funktion `fib3`, die die Fibonacci-Zahlen wie in Gleichung 3 berechnet. (2 Punkte)

  Hinweis: Am besten definiert Ihr die Indizes a und b in Python als optionale Parameter:

```
def fib3(n, a=0, b=1):
  # Hier kommt der Code!
```

- **6.1.4** Schreibt eine Python Funktion `fib4`, die die Fibonacci-Zahlen wie in Gleichung (1) rekursiv berechnet, aber sich schon einmal berechnete Werte merkt und bei erneutem Aufruf ohne weitere Berechnung zurückgibt. Benutze ein Wörterbuch zur Speicherung der Werte. (2 Punkte)

- **6.1.5** Berechnet mit Hilfe der Funktionen `fib1`, `fib2`, `fib3` und `fib4` aus den vorigen Aufgaben die Fibonacci-Zahl für $n = 20$ und $n = 40$. Was fällt Euch bei den Laufzeiten auf? Berechnet das asymptotische Laufzeitverhalten Eurer Implementationen und begründet damit das beobachtete Verhalten. (2 Punkte)

  Hinweis:

  – Zur Laufzeitmessung könnt Ihr den Shellbefehl `time` wie folgt verwenden (in der Shell, *nicht* in Python!):

  ```
  > time python fib.py
  121393

  real    0m0.080s
  user    0m0.070s
  sys     0m0.000s
  ```

  – Zum Bestimmen des asymptotischen Laufzeitverhaltens siehe zum Beispiel `http://de.wikipedia.org/wiki/Landau-Symbole`. In diesem Kontext geht es um die Frage, wie oft die Funktion für ein gegebenes n im Limes $n \to \infty$ aufgerufen wird.