

Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wölfl, Thorsten Engesser
Wintersemester 2015/2016

Universität Freiburg
Institut für Informatik

Übungsblatt 12

Abgabe: Freitag, 29. Januar 2016, 20:00 Uhr

WICHTIGE HINWEISE: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet12` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann in Dateien in diesem Unterverzeichnis erwartet.

Beachten Sie bitte bei allen Aufgaben die *Hinweise zur Bearbeitung der Übungsaufgaben* unter der folgenden URL:

<http://gki.informatik.uni-freiburg.de/teaching/ws1516/info1/wiki/hinweise.html>

Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 12.1 (timeit, Punkte: 2+3, Dateien: `test_log.py`, `test_dict.py`)

Verwenden Sie das `timeit`-Modul, um die folgenden Fragestellungen zu beantworten. Verwenden Sie dazu eines der beiden Templates, die Sie von der Webseite herunterladen, umbenennen und für die jeweilige Fragestellung adaptieren können. Jeder Test soll 3-mal wiederholt werden. Die Anzahl der Durchläufe (Loops) in jedem Testlauf legen Sie dabei so fest, dass die Gesamtlaufzeit pro Testlauf etwa zwischen 1 und 5 sec. liegt. Gestalten Sie die Ausgabe so, dass ersichtlich wird, für welchen Test die Laufzeiten gemessen wurden.

- (a) Sie wollen sehr häufig eine Funktion `spam_log` verwenden, die die Funktion `log2` aus dem Modul `math` genau 1000 mal aufruft. Welche Implementierung ist besser: (1) eine, die das Modul `math` importiert und in der Definition von `spam_log` die Logarithmusfunktion über den Namen `math.log2` aufruft oder (2) eine, die aus dem Modul `math` die Funktion `log2` importiert und in der Definition von `spam_log` die Logarithmusfunktion über den Namen `log2` aufruft?
- (b) Gegeben sei ein Dictionary `dct` mit etwa 1000 Einträgen und eine Liste von *möglichen* Keys des Dictionary. Sie wollen für jeden Eintrag `k` der Liste, sofern er ein Schlüssel von `dct` ist, auf den Wert `dct[k]` zugreifen. Welche Implementierung ist besser: (1) eine, die über die Listenelemente iteriert und vor dem Zugriff auf `dct[k]` abfragt, ob `k` ein Key von `dct` ist, und nur dann auf `dct[k]` zugreift; oder (2) eine, die den Zugriff auf `dct[k]` mittels einer `try: ... except:-`Klausel „absichert“.

Testen Sie folgende Szenarien: etwa die Hälfte der Listeneinträge sind Keys des Dictionary, kein Listeneintrag ist ein Key, alle Einträge sind Keys.

Aufgabe 12.2 (Tupel vs. Listen; Punkte: 4, Dateien: `test_tuple_list.py`)

Tuples are faster than lists. Diese Feststellung findet sich in einigen Online-Dokumentationen zu Python, aber stimmt das? Um diese Frage zu analysieren, interessiert uns:

- (a) Ist der Zugriff auf einen Listeneintrag `x = lst[k]` langsamer als der Zugriff auf einen Tupeleintrag `x = tpl[k]`?
- (b) Ist die Erzeugung eines Listen-Slices `x = lst[k:n]` langsamer als die Erzeugung eines Tupel-Slices `x = tpl[k:n]`?

Diskutieren Sie obige Fragestellung anhand eigener Testergebnisse (als Kommentar in der Datei `test_tuple_list.py`). Verwenden Sie dazu ebenfalls das `timeit`-Modul und eines der beiden bereitgestellten Templates. Gestalten Sie die Ausgabe so, dass ersichtlich wird, für welche der beiden Fragen (a), (b) und welchen Datentyp Ihre Zeitmessungen gelten.

Aufgabe 12.3 (Komplexitätsanalyse; Punkte: 3+3+3; Dateien: `spam.py`, `landau.txt`)
Betrachten Sie die Funktion `spam(c)`, die als Argument `c` ein Containerobjekt (z.B. ein `set` oder eine `list`) mit Zahlen übernimmt.

```
def spam(c):  
    eggs = 0  
    for elem in c:  
        if elem * elem in c:  
            eggs += 1  
    return eggs
```

- (a) Schätzen Sie mit Hilfe der Landauschen $\mathcal{O}$ -Notation die asymptotische Laufzeit eines Aufrufs von `spam(c)` in Abhängigkeit von der *Länge des Containers* `c` ab. Unterscheiden Sie dazu die folgenden beiden Fälle: `c` ist ein Objekt vom Typ `list` bzw. `c` ist ein Objekt vom Typ `set`.
- (b) Führen Sie dann eine empirische Analyse des Laufzeitverhaltens von `spam(c)` mittels des `timeit`-Moduls durch. Bestimmen Sie dazu die Laufzeit jeweils für Listen und Mengen mit 100, 150, 200 und 250 Elementen. Geben Sie Ihre Ergebnisse (als Kommentar in `spam.py`) tabellarisch an.
- (c) Geben Sie für die unten angegebenen Funktionen $a(n)$, $b(n)$, $c(n)$ und $d(n)$ paarweise an, in welcher asymptotischen Beziehung sie zueinander stehen. Benutzen Sie dazu die Landausche $\mathcal{O}$ -Notation und begründen Sie kurz (in der Datei `landau.txt`).

$$a(n) = 2^n + n^3, \quad b(n) = n^3 \log(2n), \quad c(n) = 2^{n+3}, \quad d(n) = 42^{1000}$$

Aufgabe 12.4 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet12` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Bezug zur Vorlesung, Interessantes, etc.).