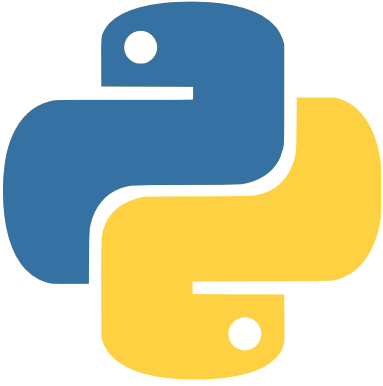
Universität Stuttgart
Projekt digit@L – BOOST. SKILLS. SUPPORT.



Programmierkurs Python

Zeitmessungen und Code-Performance

Dominik Göddeke

IANS SimTech

Stiftung Innovation in der Hochschullehre

CC BY

Zeitmessung und Code-Performance

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen
- Sowie auf den Datenstrukturen basierenden bereitgestellten Funktionalitäten

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen
- Sowie auf den Datenstrukturen basierenden bereitgestellten Funktionalitäten
- **Beispiel: Mengen auch als Listen** realisierbar

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen
- Sowie auf den Datenstrukturen basierenden bereitgestellten Funktionalitäten
- **Beispiel: Mengen auch als Listen** realisierbar
  - Eindeutiges Vorkommen „von Hand“ sicherstellen

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen
- Sowie auf den Datenstrukturen basierenden bereitgestellten Funktionalitäten
- **Beispiel: Mengen auch als Listen** realisierbar
  - Eindeutiges Vorkommen „von Hand“ sicherstellen
  - Umgekehrt: **Liste als Menge von Tupeln**

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen
- Sowie auf den Datenstrukturen basierenden bereitgestellten Funktionalitäten
- **Beispiel: Mengen auch als Listen** realisierbar
  - Eindeutiges Vorkommen „von Hand“ sicherstellen
  - Umgekehrt: **Liste als Menge von Tupeln**
  - In jedem Tupel ein Listenelement und seine Position in der Liste

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen
- Sowie auf den Datenstrukturen basierenden bereitgestellten Funktionalitäten
- **Beispiel: Mengen auch als Listen** realisierbar
  - Eindeutiges Vorkommen „von Hand“ sicherstellen
  - Umgekehrt: **Liste als Menge von Tupeln**
  - In jedem Tupel ein Listenelement und seine Position in der Liste
  - Verkünstelt die Eindeutigkeit der Elemente einer Menge

Motivation

- Zwischenstand: **viele Problemstellungen** auf **verschiedene Arten lösbar**
- Insbesondere: mit verschiedenen Datenstrukturen
- Sowie auf den Datenstrukturen basierenden bereitgestellten Funktionalitäten
- **Beispiel: Mengen auch als Listen** realisierbar
  - Eindeutiges Vorkommen „von Hand“ sicherstellen
  - Umgekehrt: **Liste als Menge von Tupeln**
  - In jedem Tupel ein Listenelement und seine Position in der Liste
  - Verkünstelt die Eindeutigkeit der Elemente einer Menge
  - Wenn funktional equivalent, warum dann beide?

Motivation

- **Kunst der Programmierung**

Motivation

- **Kunst der Programmierung**
- Im Wesentlichen eine Frage der Erfahrung und Übung

Motivation

- **Kunst der Programmierung**
- Im Wesentlichen eine Frage der Erfahrung und Übung
- **Auswahl einer geeigneten Datenstruktur**, und **Auswahl der „besten“ Operationen** auf der Datenstruktur

Motivation

- **Kunst der Programmierung**
- Im Wesentlichen eine Frage der Erfahrung und Übung
- **Auswahl einer geeigneten Datenstruktur**, und **Auswahl der „besten“ Operationen** auf der Datenstruktur
- Beides offenbar **stark problemabhängig**

Motivation

- **Kunst der Programmierung**
- Im Wesentlichen eine Frage der Erfahrung und Übung
- **Auswahl einer geeigneten Datenstruktur**, und **Auswahl der „besten“ Operationen** auf der Datenstruktur
- Beides offenbar **stark problemabhängig**
- Generelle Frage, wie Programme **möglichst optimal** sind

Motivation

- **Kunst der Programmierung**
- Im Wesentlichen eine Frage der Erfahrung und Übung
- **Auswahl einer geeigneten Datenstruktur**, und **Auswahl der „besten“ Operationen** auf der Datenstruktur
- Beides offenbar **stark problemabhängig**
- Generelle Frage, wie Programme **möglichst optimal** sind
- Zu Beginn der Programmierausbildung:

Motivation

- **Kunst der Programmierung**
- Im Wesentlichen eine Frage der Erfahrung und Übung
- **Auswahl einer geeigneten Datenstruktur**, und **Auswahl der „besten“ Operationen** auf der Datenstruktur
- Beides offenbar **stark problemabhängig**
- Generelle Frage, wie Programme **möglichst optimal** sind
- Zu Beginn der Programmierausbildung:
  - Fehlende Erfahrung

Motivation

- **Kunst der Programmierung**
- Im Wesentlichen eine Frage der Erfahrung und Übung
- **Auswahl einer geeigneten Datenstruktur**, und **Auswahl der „besten“ Operationen** auf der Datenstruktur
- Beides offenbar **stark problemabhängig**
- Generelle Frage, wie Programme **möglichst optimal** sind
- Zu Beginn der Programmierausbildung:
  - Fehlende Erfahrung
  - Kein Wissen „gute“ Entscheidung a priori zu treffen

Idee: Experimentelle Erfahrung auf der Basis von Laufzeiten

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“
- Laufzeit: Zeit, die der Computer tatsächlich benötigt, um unseren Code auszuführen, und nicht eine irgendwie geratene oder begründete Erwartung

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“
- Laufzeit: Zeit, die der Computer tatsächlich benötigt, um unseren Code auszuführen, und nicht eine irgendwie geratene oder begründete Erwartung
- Dazu **Python-Bordmittel**

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“
- Laufzeit: Zeit, die der Computer tatsächlich benötigt, um unseren Code auszuführen, und nicht eine irgendwie geratene oder begründete Erwartung
- Dazu **Python-Bordmittel**
  - Importieren des Moduls `time`

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“
- Laufzeit: Zeit, die der Computer tatsächlich benötigt, um unseren Code auszuführen, und nicht eine irgendwie geratene oder begründete Erwartung
- Dazu **Python-Bordmittel**
  - Importieren des Moduls `time`
  - Darin: Funktion `perf_counter()`

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“
- Laufzeit: Zeit, die der Computer tatsächlich benötigt, um unseren Code auszuführen, und nicht eine irgendwie geratene oder begründete Erwartung
- Dazu **Python-Bordmittel**
  - Importieren des Moduls `time`
  - Darin: Funktion `perf_counter()`
  - Messung der Laufzeit einer Anweisung analog zu Stopuhr

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“
- Laufzeit: Zeit, die der Computer tatsächlich benötigt, um unseren Code auszuführen, und nicht eine irgendwie geratene oder begründete Erwartung
- Dazu **Python-Bordmittel**
  - Importieren des Moduls `time`
  - Darin: Funktion `perf_counter()`
  - Messung der Laufzeit einer Anweisung analog zu Stopuhr
  - Empfehlung: viele Durchläufe, und Mittelung, **Zeitmessungen immer verrauscht**

Experimentelle Erfahrung auf der Basis von Laufzeiten

- Offensichtlich: **Laufzeit** ist eine gute „Metrik“
- Laufzeit: Zeit, die der Computer tatsächlich benötigt, um unseren Code auszuführen, und nicht eine irgendwie geratene oder begründete Erwartung
- Dazu **Python-Bordmittel**
  - Importieren des Moduls `time`
  - Darin: Funktion `perf_counter()`
  - Messung der Laufzeit einer Anweisung analog zu Stopuhr
  - Empfehlung: viele Durchläufe, und Mittelung, **Zeitmessungen immer verrauscht**
  - Nennt man **Profiling**

Codebeispiele

Beispiel: Zugriff auf Listen und Mengen

Diskussion

Diskussion

- Bemerkenswert großer Unterschied (!)

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit
- Hier wichtig

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit
- Hier wichtig
  - Berücksichtigung von a priori Wissen im **Entwurf von Programmen**

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit
- Hier wichtig
  - Berücksichtigung von a priori Wissen im **Entwurf von Programmen**
  - Welche Datenstruktur „passt“?

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit
- Hier wichtig
  - Berücksichtigung von a priori Wissen im **Entwurf von Programmen**
  - Welche Datenstruktur „passt“?
  - Nicht nur im Bezug auf Funktionalität, sondern auch im Bezug auf Laufzeit

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit
- Hier wichtig
  - Berücksichtigung von a priori Wissen im **Entwurf von Programmen**
  - Welche Datenstruktur „passt“?
  - Nicht nur im Bezug auf Funktionalität, sondern auch im Bezug auf Laufzeit
  - Wenn wir keine Ahnung haben: Laufzeit-Experimente verstärken Intuition

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit
- Hier wichtig
  - Berücksichtigung von a priori Wissen im **Entwurf von Programmen**
  - Welche Datenstruktur „passt“?
  - Nicht nur im Bezug auf Funktionalität, sondern auch im Bezug auf Laufzeit
  - Wenn wir keine Ahnung haben: Laufzeit-Experimente verstärken Intuition
  - Ohne Detailkenntnis „unter der Haube“

Diskussion

- Bemerkenswert großer Unterschied (!)
- Python „unter der Haube“ schlauer bei Eindeutigkeit
- Hier wichtig
  - Berücksichtigung von a priori Wissen im **Entwurf von Programmen**
  - Welche Datenstruktur „passt“?
  - Nicht nur im Bezug auf Funktionalität, sondern auch im Bezug auf Laufzeit
  - Wenn wir keine Ahnung haben: Laufzeit-Experimente verstärken Intuition
  - Ohne Detailkenntnis „unter der Haube“
- Fazit: beruhigend, verschiebt Verantwortung auf uns als Programmierer*innen

Impressum, Danksagung und Quellen

Stiftung Innovation in der Hochschullehre

Gefördert durch die Stiftung Innovation in der Hochschullehre im Rahmen des Projekts digit@L, https://stiftung-hochschullehre.de

SimTech

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie

---

Autor: Dominik Göddeke, IANS, Universität Stuttgart

CC BY

Weitere Quellen:

- Logos Universität Stuttgart, IANS, SimTech: Universität Stuttgart, alle Rechte vorbehalten
- Logo Python: https://freesvg.org/387, CC-0
- Logo Stiftung: Stiftung Innovation in der Hochschullehre, alle Rechte vorbehalten
- Logo ZOERR: Universität Tübingen, alle Rechte vorbehalten

ZOERR

Veröffentlicht auf dem Zentralen OER Repositorium Baden-Württemberg, https://www.zoerr.de