Computerlinguistik I

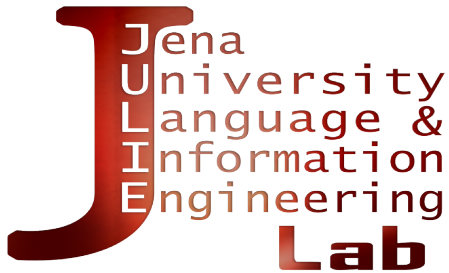
Übung zur Vorlesung

Sven Büchel

Jena Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena, Germany

https://julielab.de/

Wintersemester 2018/19



Organisatorisches

Algorithmik und Programmierung

- Einführung
- Variablen

Abschnitt 1

Organisatorisches

Vorstellungsrunde

- Name
- Fach
- Semester
- Programmierkenntisse?
- Erwartungen?
- Verhältnis zum restlichen Curriculum?

Kontaktdaten

- sven.buechel@uni-jena.de
- https://julielab.de/Staff/Buechel/
- +49 3641 9 44324
- Fürstengraben 27, E 009

Aufbau und Ziele der Übung

- Klärung offener Fragen(!) zum Vorlesungsstoff
- Vertiefte Auseinandersetzung mit formalen Konzepten
- Vorstellung und Nachbesprechung der Übungsblätter

Übungsblätter

- Wöchentlich zu lösende Aufgaben
- 50% für Prüfungszulassung
- `https://julielab.de/Students.html`
- Abzugeben als XXX
- Per Email an `sven.buechel@uni-jena.de`
- Jeweils bis 23:59 am XXX
- Angabe von Name, Matrikel-Nummer, Datum, Veranstaltung
- Bitte Schwarz-Weiß!
- Macht etwa ein Drittel der Gesamtnote aus (nur Verbesserung möglich)
- Gruppenarbeit ist explizit zugelassen, muss aber kenntlich gemacht werden (ein Lösungsblatt mit allen Namen)

Ablauf der Sitzungen

- Organisatorisches
- Letztes Übungsblatt besprechen
- Vorlesung nachbesprechen (studentengetrieben!)
- Ggf. zusätzliche Übungen während der Sitzung
- Vorstellung des nächsten Übungsblatts

Ablauf des Semesters

- Algorithmik und Programmierung
- Formale Konzepte der Vorlesung (Automaten, Transduktoren, Formale Sprachen, Grammatiken, Syntaxbäume/Parsing)
- Zusammenfassung und Prüfungsvorbereitung

Verlegung des Übungstermins

- Verlegung nur bei einstimmigem Votum
- Neue Ort wäre Fürstengraben 27, Raum E12 (Computerkabinett)
- Doodle-Link:
  https://doodle.com/poll/pu5k8vq2nduztq8p

Technischer Rahmen der Programmierübungen I

Dieses Semester wird in der Übung erstmalig konkreter Programmcode statt abstrakter Pseudocode verwendet. Es gibt unterschiedliche Möglichkeiten, diesen Code auszuführen.

- Lokal installierte Software
  `https://conda.io/docs/user-guide/install/index.html`
  - Pro: unabhängig von Drittanbietern, Vermittlung technischer Kenntnisse, offline-Arbeiten möglich
  - Kontra: technischer Schwierigkeitsgrad, unterschiedliche Lösungen je nach Rechner

Technischer Rahmen der Programmierübungen II

Dieses Semester wird in der Übung erstmalig konkreter Programmcode statt abstrakter Pseudocode verwendet. Es gibt unterschiedliche Möglichkeiten, diesen Code auszuführen.

- “normale” Web-basierte Python-Interpreter
  Z.B. `https://repl.it`
  - Pro: Keine lokale Installation nötig
  - Kontra: Abhängigkeit von (verschiedenen Drittanbietern), kein offline-Arbeiten

Technischer Rahmen der Programmierübungen III

Dieses Semester wird in der Übung erstmalig konkreter Programmcode statt abstrakter Pseudocode verwendet. Es gibt unterschiedliche Möglichkeiten, diesen Code auszuführen.

- Google Colab
  `https://colab.research.google.com/`
  - Pro: Integration von Textblöcken und Code in sog. Jupyter Notebooks, keine lokale Installation, Austausch von Aufgaben und Lösungen über Google Drive
  - Kontra: Alle brauchen Google Accounts, kein offline-Arbeiten

Literaturhinweise

- Jurafsky & Martin. Speech and Language Processing
  - 2. Auflage (Lehrbuchsammlung)

    `https://kataloge.thulb.uni-jena.de/DB=1/SET=2/TTL=1/SHW?FRST=1`
  - 3. Auflage (online preprint)

    `https://web.stanford.edu/~jurafsky/slp3/`

Abschnitt 2

Algorithmik und Programmierung

Unterabschnitt 1

Einführung

Algorithmen

- Abfolge *gültiger* Anweisungen
- I.d.R. zur Lösung eines Problems. Z.B. ...
  - Wortarten erkennen
  - Syntaxbaum berechnen
  - Emotion eines Satzes bestimmen
- I.d.R zur Ausführung durch Computer bestimmt
  - Gegenbeispiel: schriftliches Addieren

Programme

- Abfolge gültiger Anweisungen einer konkreten Programmiersprache (z.B. C, Java, Python) (Quellcode)
  Implementierung eines abstrakten Algorithmus
- Durch Computer ausführbar
- Derselbe Algorithmus kann in unterschiedlichen Programiersprachen implementiert werden
- In diesem Kurs nutzen wir **Python** 3

Hello, World!

```
1  print('Hello, World!')  # Ausgabe: Hello, World!
```

- Der `print`-Befehl erzeugt eine Ausgabe auf dem Bildschirm
- Das Doppelkreuz '#' markiert **Kommentare**. Dieses und alles rechts davon in einer Zeile wird vom **Interpreter** ignoriert.

Sequenzielle Bearbeitung

Ein Programm wird von oben nach unten, Zeile für Zeile ausgeführt.

```
1  print('Hello, World!')
2  print(42)
3  print(2+2)
```

Ausgabe:

```
Hello, World!
42
4
```

Unterabschnitt 2

Variablen