Computerlinguistik I

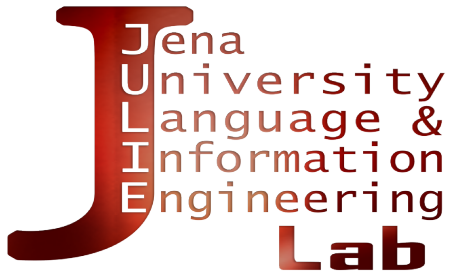
Übung zur Vorlesung

Sven Büchel

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

https://julielab.de/

Wintersemester 2018/19



Organisatorisches

Algorithmik und Programmierung

- Einleitung
- Variablen
- Datentypen und Operationen
- Kontrollstrukturen
- Funktionen
- Rekursion

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 Jupyter Notebook (.ipynb), später ggf. .zip
- Per Email an `sven.buechel@uni-jena.de`
- Jeweils bis Sonntag, 23:59
- Macht etwa ein Drittel der Gesamtnote aus (nur Verbesserung möglich)
- Gruppenarbeit ist explizit zugelassen, muss aber kenntlich gemacht werden (ein Lösungsblatt mit allen Namen). Max. 3 Personen pro Gruppe.

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

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

Einleitung

Ziele

- Kein Programmierkurs
  (keine Anwendungsentwicklung, nur minimale technischen Hintergründe, ...)
- Keine Einführung in die Informatik
- Lesen und Schreiben von Algorithmen
- Lauffähiger Code
- Grundlage für Verständnis computerlinguistischer Methoden

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

Variablen im Mathematikunterricht

- Variablen sind sind Symbole, die verschiedene Werte einnehmen können
- $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2 + 5$
- Zuweisung verbal z.B. “Sei x gleich 7”.

Variablen in der Informatik

- Variablen als “benamte Zeiger”
- Referenzieren Werte im Speicher
- Zugriff über den Variablennamen

```
1  # Variablenzuweisungen
2  x = 5
3  y = 7
4  z = 8
5  print(y)   # Ausgabe 7
6  # Rechnen mit Variablen
7  x = 3 * z
8  x = x / 2
9  z = x + y
10 print(z)   # Ausgabe 19
```

Werte und Variablen

Variablennamen

- Dürfen Großbuchstaben, Kleinbuchstaben, Zahlen und den Unterstrich enthalten (keine Sonderzeichen, kein Leerzeichen)
- Müssen mit Buchstabe oder Unterstrich anfangen (i.d.R klein)
- Dürfen ansonsten beliebig ausgewählt werden
- Sollten knapp und präzise bezeichnen, was referenziert wird
  - gut: `data`, `results`, `input_text`
  - weniger gut: `dskjfhal`, `var7`, `Baumhaus`
- Mehrere Wörter werden per Konvention mit Unterstrich (`variable_name`) oder Binnenmajuskel (`variableName`) getrennt

Wertezuweisung

```
1  # Variablen wird mit "=" Werte zugewiesen
2  x = 5
3
4  # Dies darf jedoch nicht als Gleichheit
       verstanden werden, wie folgedes Beispiel
       zeigt
5  x = x + 1   # Der Wert von x wird um 1 erhoeht
6  print(x)    # Ausgabe 6
```

Unterabschnitt 3

Datentypen und Operationen

Motivation

- In der Schule nehmen Variablen meist nur Zahlenwerte (ganze, reelle, komplexe, usw.)
- In der Computerlinguistik verarbeiten wir jedoch Sprache und brauchen daher (auch) andere “Arten von Werten” (**Datentypen**)
- Datentypen unterscheiden sich hinsichtlich ...
  - des erlaubten Wertebereichs (“Haus” ist keine Zahl) und
  - der erlaubten Operationen (4 + 3 vs. 4 + “Haus”)

Für die Übung relevante Datentypen

- Wahrheitswerte (`booleans`)
- Zahlen (`integer, floats`)
- Zeichenketten (`strings`)
- Listen (`lists`)
- `dictionaries`

Anders als in vielen anderen Programmiersprachen werden Datenypen in Python nicht explizit festgelegt und sind veränderlich (dynamisches Typensystem).

```
1  var1 = 5   # Java: int var1 = 5;
2  var2 = 7
3  print(var1 + var2)   # Ausgabe 12
4  var1 = "Haus"   # geaenderter Variablentyp
5  print(var1 + var2)   # Fehler
```

Wahrheitswerte (Booleans)

- Nehmen entweder den Wert `True` oder `False` an
- Aussagenlogische Operatoren:
  - und (`and`)
  - oder (`or`)
  - nicht (`nicht`)

Wahrheitswerte (Booleans) II

```
1 var1 = True
2 var2 = False
3 print(var1)
4 print(var1 and var2)
5 print(var1 or var2)
6 print(not var1)
7 print(not (var1 or var2))
```

Exkurs: Aussagenlogik

Semantik (Bedeutung) der aussagenlogischen Operatoren in Form einer Wahrheitswert-Tabellen

a	b	`not` a	a `or` b	a `and` b
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

Exkurs: Aussagenlogik II

- Wahrheitswert-Tabelle zur Bestimmung des Wahrheitswerts komplexer Aussagen in Abhängigkeit des Wahrheitswerts von Elementaraussagen (logischer Atome) an

a	b	`not` a	(`not` a) `or` b
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

Exkurs: Übung zur Aussagenlogik

Ermitteln Sie den Wahrheitswertverlauf der komplexen Aussagen “(a `and` b) `or` b” mithilfe einer Wahrheitswerttabelle.

Exkurs: Übung zur Aussagenlogik (Lösung)

a	b	a `and` b	(a `and` b) `or` b
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	1

Zahlen

- Die wichtigsten Zahlentypen in Python sind `Integer` (Ganzzahlen) und `Floats` (Fließkommazahlen)
- Seit Python 3 ist der Unterschied zwischen beiden für den Einsteiger eher irrelavant
- Grundrechenarten (`+ - * /`) funktionieren jetzt für beide Datentypen im Ergebnis gleich

```
1  x = 9   # Integer
2  y = 4   # Integer
3  print(x / y) # Ausgabe 2.25 (Float)
4  # In Python 2 waere hier '2' ausgegeben worden.
5  x = 9.0   # Float. Achtung, Punkt statt Komma!
6  y = 4.0   # Float
7  print(x / y) # Ausgabe 2.25 (Float)
```

Vergleichsoperatoren

- Werden zu Booleans ausgewertet
- Gleicheit ==
- Kleiner <
- Größer >
- Kleinergleich <=
- Größergleich >=

```
1   print(4 == 4)
2   print(4 > 4)
3   print(4 <= 4)
```

Einige weiterführende Zahlenoperatoren

```
1  # Potenzen
2  print(4**4) # Ausgabe 256
3
4  # Zuweisung mit Addition
5  x = 4
6  x += 1   # wie x = x + 1
7  print(x)
8  # genauso -=, *= und /=
9
10 # Division mit Rest
11 print(9 % 8)
```

Zeichenketten (Strings)

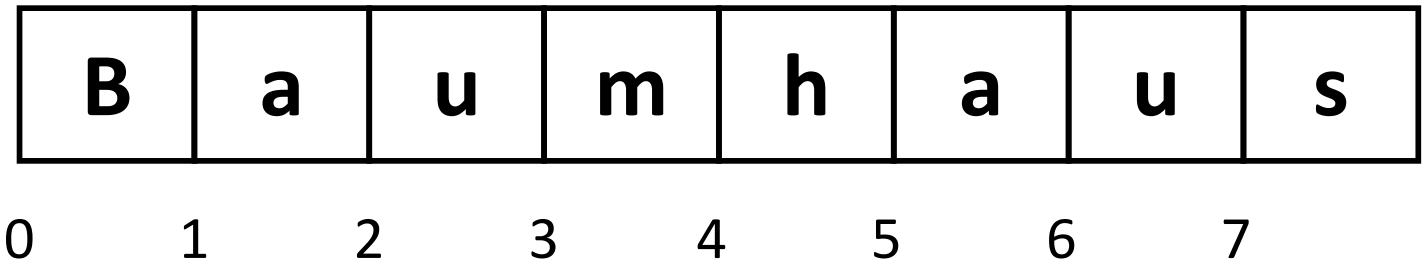
```
1  # Strings werden mit Anfuehrungszeichen
       ausgezeichnet (doppelt oder einfach)
2  x = "Baum"
3
4  # Strings, die Zahlen enthalten sind von den
       Zahlwerten an sich zu unterscheiden
5  x = "8"
6  y = 9
7  print(x + y) # Fehler
```

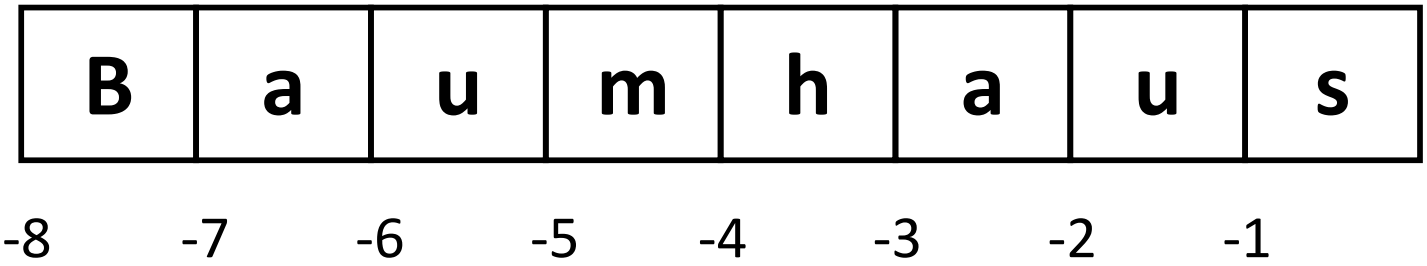
Verarbeiten von Strings

```
1 # Konkatenation (Verbinden) von Strings
2 print("Baum" + "Haus")
3
4 # Laengen-Funktion
5 a = "Baumhaus"
6 b = len(a)
7 print(b)
```

Strings: Indexzugriff

- Strings sind Folgen von einzelnen Zeichen
- Auf diese kann durch den jeweiligen **Index** (“Stellenbezeichner”) einzeln zugegriffen werden
- Vorwärtsindexierung (beginnend mit 0) vs. Rückwärtsindexierung (beginnend mit -1)





Strings: Indexzugriff

- Beim Indexzugriff wird durch die Syntax `string_name[n]` auf das *n*-te Zeichen des Strings zugegriffen
- Für nachfolgende Stringoperationen (insbesondere Slicing) hilft es sich vorzustellen, dass die Indizes “zwischen” den Stellen stehen und beim normalen Indexzugriff immer das Element rechts davon abgerufen wird (gilt für Vorwärts- und Rückwärtsindexierung)

```
1   x = "Computerlinguistik"
2   print(x[0])   # "C"
3   print(x[8])   # "l"
4   print(x[-1])   # "k"
```

Übung zur Stringindexierung

Schreiben Sie ein Programm, dass den String `Computerlinguistik ist toll!"` in der Variable `Meinung` speichert. Erzeugen Sie die Variablen `a, b, c, d` die jeweils das erste Zeichen der Worte in `Meinung` sowie das Interpunktionszeichen speichert. Anschließend soll das Programm den Wert von `Meinung` mit dem String `"Wirklich!"` konkatenieren und dies dann ausgeben. Stellen Sie sicher, dass das ausgegebene Endergebnis orthografisch korrekt ist.

Übung zur Stringindexierung (Lösung)

```
1 Meinung = "Computerlinguistik ist toll!"
2 a = Meinung[0]
3 b = Meinung[19]
4 c = Meinung[23]
5 d = Meinung[-1]
6 #print(a,b,c,d)
7 print( Meinung + " Wirklich!")
```