Computerlinguistik I

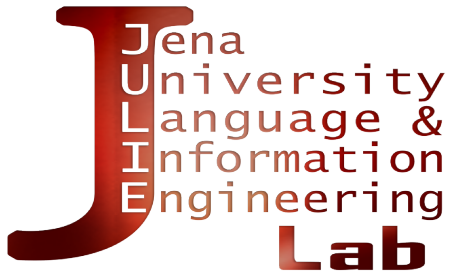
Übung zur Vorlesung

Sven Büchel

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

https://julielab.de/

Wintersemester 2018/19



Zeichenketten (Strings)

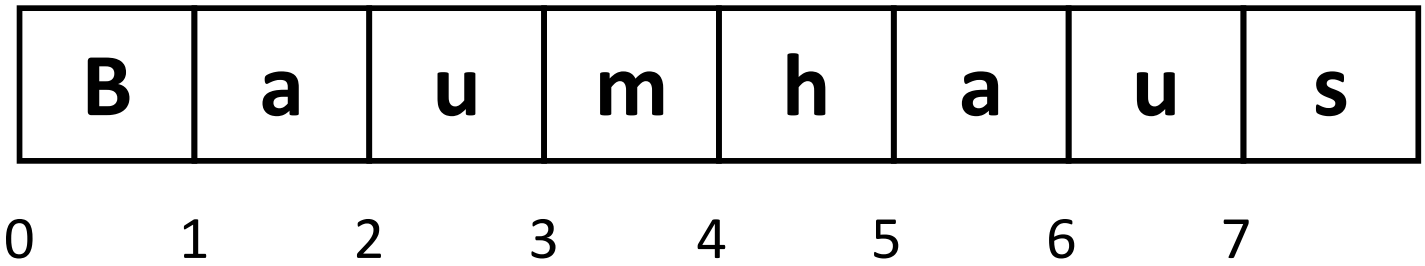
```
1  # Strings werden mit Anfuehrungszeichen
       ausgezeichnet (doppelt oder einfach)
2  x = "Baum"
3
4  # Strings, die Zahlen enthalten sind von den
       Zahlwerten an sich zu unterscheiden
5  x = "8"
6  y = 9
7  print(x + y) # Fehler
```

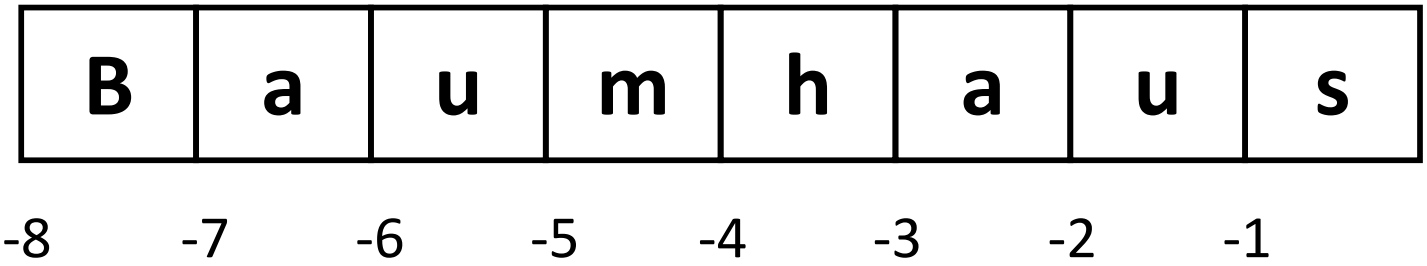
Verarbeiten von Strings

```
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!")
```

Strings: Slicing

- Um längere Teilketten (Substrings) abzurufen, kann man sogenanntes **Slicing** verwenden.

```
1  some_string = "Computerlinguistik"
2
3  # Allgemeine Syntax: string[beginIndex:endIndex]
4  print(some_string[1:4]) # "omp"
5
6  # Der erste bzw. letzt Index kann weggelassen
       werden, wenn der Teilstring vom Anfang bzw.
       bis zum Ende geht.
7  print(some_string[:8]) # Computer
8  print(some_string[8:]) # linguistik
```

Listen

- Eine Liste ist eine Datenstruktur, die mehrere unterschiedliche Werte verschiedenen Datentyps speichern kann
- Syntaktisch durch eckige Klammer `[]` markiert. Einzelne Elemente werden mit Kommata abgegrenzt.
- Listen können Listen enthalten!
- Erlaubte Operationen weitgehend identisch zu String (Konkatenation, `len`, Indexzugriff, Slicing)
- Listen können nur mit Listen (nicht einzelnen Elementen) konkateniert werden. Um einzelne Elemente anzuhängen, kann die `append`-Methode verwendet werden (s. nächste Folie).

Erzeugen und Manipulieren von Listen

```
1  l = [] # leere Liste initialisieren
2  l = [1, "1", True, "Computerlinguistik"] # Liste
       mit Elementen initialisieren
3  print(l[0]) # 1
4  print(l[1]) # "1"
5
6  l = l + ["ist", "toll"] # Konkatenation von
       Listen
7  l.append("!") # Syntax zum anhaengen einzelner
       Elemente
8  print(l[-1]) # "!"
9
10 # Slicing von Listen
11 print(l[-4:]) # ["Computerlinguistik", "ist", "
       toll", "!"]
```

Listen — Achtung: Referenzkopie vs. Wertekopie

```
1  list1 = [1,2,3]
2  list2 = list1
3  list2.append(4)
4  print(list1[-1])   # 4 !!!
```

Übung zu Slicing und Listen

Schreiben Sie ein Programm, dass den String `Computerlingustik ist toll"` in einer Variable speichert. Erzeugen Sie eine leere Liste. Verwenden Sie Slicing, um die einzelnen Wörter im obigen Satz als einzelne Listenelemente zu speichern. Lassen Sie die Länge der Liste ausgeben.

Übung zu Slicing und Listen (Lösung)

```
1  s = "Computerlinguistik ist toll"
2  liste = [s[:18], s[19:22], s[23:]]
3  # print(liste)
4  print(len(liste))
```

Zerlegungen und Verbinden von Strings bzw. Listen

```
1 # split-Methode: Listen aus Strings erzeugen
2 s = "Dies ist ein Satz"
3 l = s.split(" ")
4 print(l) # ["Dies", "ist", "ein", Satz"]
5
6 # join-Methode um Strings aus Listen
     zusammenzufuegen
7 l = ["Dies", "ist", "ein", "Satz"]
8 s = " ".join(l)
9 print(s)  # "Dies ist ein Satz"
```

Dictionaries

- Menge (ungeordnet) von Schlüssel-Wert-Paaren (Key-Value)
- Angabe des Schlüssel erlaubten direkten Zugriff auf Wert
- Häufig werden `int`- und `String`-Datentypen als Schlüssel verwendet

```
1  # Neues dictionary erstellen
2  d = {1: "Mo", 2:" Di", 3: "Mi", 4: "Do",
3      5: "Fr", 6: "Sa", 7: "So"}
4
5  # Wert durch einen Schluessel abrufen
6  print(d[1])       # Montag
7  print(d.get(1))   # Alternative, gibt None wenn
       Schluessel nicht enthalten
8
9  # Neues Schluessel-Wert-Paar hinzufuegen
10 d[8] = "Fanatsietag"
```

Dictionaries — Linguistische Beispiele

```
1  # Beispiel mentales Lexikon
2  lexicon = {"werfen": {"1sg": "werfe",
3                        "2sg": "wirfst"},
4             "gehen": {"1sg": "gehe",
5                       "2sg": "gehst"}
6             }
7  print("ich " + lexicon["werfen"]["1sg"])  # "ich
       werfe"
8
9  # Beispiel haeufigste Wortart
10 most_freq_pos = {"der": "det",
11                  "ein": "det",
12                  "Baum": "NN",
13                  "fahren": "V"}
```