DS 10 – Zeichenketten: Texte Zeichen für Zeichen

Informatik 10 – MINT-Zweig | Datenstrukturen

Lernziele

Ich kann

- eine Zeichenkette als Folge (Sequenz) von Zeichen beschreiben und den Index eines Zeichens angeben,
- mit `text[index]` auf einzelne Zeichen zugreifen und mit `len(text)` die Länge bestimmen,
- Zeichenketten verketten und Ausgaben mit f-Strings bequem zusammensetzen,
- mit einer Schleife alle Zeichen einer Zeichenkette durchlaufen,
- `in`, `.upper()`, `.lower()`, `.isalpha()` und `.isdigit()` anwenden,
- einen `IndexError` erklären und vermeiden.

🗣️ Sprachhilfe

- „Das Zeichen am Index … ist …, weil ab 0 gezählt wird."
- „Die Schleife durchläuft nacheinander alle Zeichen. In jedem Durchlauf prüft sie, ob …, und erhöht dann …"
- „Der `IndexError` entsteht, weil der Text nur … Zeichen hat. Gültig sind die Indizes 0 bis … ."

Worum es geht

Namen, Passwörter, Nachrichten: Sehr viele Daten in Programmen sind Texte. Bisher hast du Texte nur als Ganzes benutzt. Oft muss ein Programm aber **in einen Text hineinschauen**: Wie lang ist das Passwort? Enthält es eine Ziffer? Mit welchem Buchstaben beginnt der Name? Dafür lernst du heute, wie Python einen Text sieht: als Reihe einzelner Zeichen, jedes an einer festen Position.

1 Eine Zeichenkette ist eine Folge von Zeichen

Für Python besteht ein Text aus einzelnen **Zeichen**, die der Reihe nach hintereinander stehen, wie Perlen auf einer Schnur. Ein Zeichen ist ein Buchstabe, eine Ziffer, ein Satzzeichen oder auch ein Leerzeichen. Die Position eines Zeichens heißt **Index**. Gezählt wird ab **0**:

Index	0	1	2	3	4	5	6	7	8	9
Zeichen	`I`	`n`	`f`	`o`	`r`	`m`	`a`	`t`	`i`	`k`

Die Zeichenkette `"Informatik"` hat 10 Zeichen, aber der größte Index ist 9. Man kann sich den Index als Abstand vom Anfang vorstellen: Das erste Zeichen ist 0 Schritte vom Anfang entfernt.

> 📌 **Merksatz:** Eine **Zeichenkette** (`str`) ist eine **Sequenz von Zeichen**, z. B. `"Informatik"`. Jedes Zeichen hat eine **Position (Index)**, beginnend bei `0`.

Auf einzelne Zeichen zugreifen

Den Index schreibst du in **eckigen Klammern** hinter die Zeichenkette. Vergleiche die Ausgabe mit der Tabelle.

```python
text = "Informatik"
print(text[0])
print(text[2])
print(text[9])
```

```
I
f
k
```

`text[2]` ist das **dritte** Zeichen, `f`. In den eckigen Klammern darf auch eine Variable oder Rechnung stehen, zum Beispiel `text[i]` oder `text[i + 1]`.

Die Länge einer Zeichenkette

Wie viele Zeichen ein Text hat, verrät die Funktion `len` (englisch *length*, „Länge“). Auch das Leerzeichen zählt mit. `""` ist die **leere Zeichenkette** mit der Länge 0.

```python
text = "Informatik"
print(len(text))
print(len("Ada Lovelace"))
print(len(""))

letzter_index = len(text) - 1
print(letzter_index)
print(text[letzter_index])
print(text[-1])
```

```
10
12
0
9
k
k
```

Weil ab 0 gezählt wird, hat das **letzte Zeichen** immer den Index `len(text) - 1`. Kürzer geht es mit einem **negativen Index**, der von hinten zählt: `text[-1]` ist das letzte Zeichen, `text[-2]` das vorletzte.

> 📌 **Merksatz:** Die **Länge** erhält man mit `len(text)`. Das letzte Zeichen hat den Index `len(text) - 1` oder kurz `-1`.

⚠️ Stolperstein: Der Index ist zu groß

Führe die Zelle aus und lies die Fehlermeldung.

```python
text = "Informatik"
print(text[10])
```

```
Traceback (most recent call last):
  File "<Zelle 3>", line 2, in <module>
IndexError: string index out of range
```

`IndexError: string index out of range` heißt: „Der Index liegt außerhalb des gültigen Bereichs.“ Gültig sind bei 10 Zeichen nur die Indizes 0 bis 9 (und −1 bis −10). Prüfe bei diesem Fehler: Wie lang ist der Text, und welcher Index wurde benutzt?

🔮 Übung 1 – Zeichen bestimmen

Gib **ohne** Computer an, was jede Zeile ausgibt. Eine Zeile führt zu einem Fehler: welche, und warum? Schreibe deine Vorhersage in die Textzelle und führe den Code erst danach aus.

```python
wort = "Python"
print(wort[1])
print(len(wort))
print(wort[len(wort) - 1])
print(wort[-2])
print(wort[6])
```

```
y
6
n
o
```

```
Traceback (most recent call last):
  File "<Zelle 4>", line 6, in <module>
IndexError: string index out of range
```

Meine Vorhersage:

…

2 Zeichenketten verketten

Mit `+` hängst du Zeichenketten aneinander. Verkettung und Index-Zugriff lassen sich kombinieren.

```python
vorname = "Ada"
nachname = "Lovelace"
name = vorname + " " + nachname
print(name)
initialen = vorname[0] + nachname[0]
print(initialen)
```

```
Ada Lovelace
AL
```

Ohne das `" "` in der Mitte stünde `AdaLovelace` da. Ada Lovelace gilt übrigens als erste Programmiererin der Welt.

🧰 Werkzeug: f-Strings

Texte mit Zahlen mit `+` zusammenzusetzen ist mühsam: Jede Zahl muss mit `str(...)` umgewandelt werden. Bequemer ist ein **f-String**: ein `f` direkt vor dem ersten Anführungszeichen, und im Text Variablen oder Rechnungen in **geschweiften Klammern** `{...}`. Zeile 3 und Zeile 4 erzeugen dieselbe Ausgabe.

```python
name = "Alex"
alter = 15
print(f"{name} ist {alter} Jahre alt.")
print(name + " ist " + str(alter) + " Jahre alt.")
print(f"{name} hat {len(name)} Buchstaben.")
```

```
Alex ist 15 Jahre alt.
Alex ist 15 Jahre alt.
Alex hat 4 Buchstaben.
```

Vergisst du das `f`, erscheinen die geschweiften Klammern unverändert in der Ausgabe.

> 📌 **Merksatz:** Die **Verkettung** (Konkatenation) zweier Zeichenketten erfolgt mit `+`. Ein **f-String** `f"... {wert} ..."` setzt Werte direkt in einen Text ein.

3 Alle Zeichen durchlaufen

Mit einer `for`-Schleife bearbeitest du nacheinander **jedes Zeichen**. Statt `range(...)` schreibst du hinter `in` einfach die Zeichenkette.

```
for zeichen in "Bit":
    print(zeichen)
```

```
B
i
t
```

Die Schleifenvariable `zeichen` bekommt im ersten Durchlauf `"B"`, dann `"i"`, dann `"t"`. Der Name `zeichen` ist frei gewählt.

Zeichen zählen

🔮 **Vorhersage:** Welche Zahl gibt das Programm aus? Achte auf das große `A`.

```
1 wort = "Anna"
2 anzahl = 0
3 for zeichen in wort:
4     if zeichen == "n":
5         anzahl = anzahl + 1
6 print("Anzahl n:", anzahl)
```

```
Anzahl n: 2
```

- **Zeile 2:** Die Zählvariable `anzahl` startet bei 0.
- **Zeilen 3–5:** Die Schleife läuft über alle vier Zeichen. Nur wenn das aktuelle Zeichen ein `n` ist, wird `anzahl` um 1 erhöht.
- **Zeile 6:** Erst nach der Schleife (nicht eingerückt) wird das Ergebnis ausgegeben.

Das große `A` am Anfang wird **nicht** gezählt, denn `"A"` und `"a"` sind für Python verschiedene Zeichen.

Schritt	Zeile	Anweisung	`zeichen`	`anzahl`
1	1	`wort = "Anna"`	–	–
2	2	`anzahl = 0`	–	`0`
3	3	`for zeichen in wort:`	`"A"`	`0`
4	4	`if zeichen == "n":`	`"A"`	`0`
5	3	`for zeichen in wort:`	`"n"`	`0`
6	4	`if zeichen == "n":`	`"n"`	`0`
7	5	`anzahl = anzahl + 1`	`"n"`	`1`
8	3	`for zeichen in wort:`	`"n"`	`1`
9	4	`if zeichen == "n":`	`"n"`	`1`
10	5	`anzahl = anzahl + 1`	`"n"`	`2`
11	3	`for zeichen in wort:`	`"a"`	`2`
12	4	`if zeichen == "n":`	`"a"`	`2`
13	3	`for zeichen in wort:`	`"a"`	`2`
14	6	`print("Anzahl n:", anzahl)`	`"a"`	`2`

Einen neuen Text aufbauen

Man beginnt mit der leeren Zeichenkette `""` und hängt in jedem Durchlauf etwas an. 🔮 **Vorhersage:** Was kommt heraus?

```
neu = ""
for zeichen in "Bit":
    neu = neu + zeichen + zeichen
print(neu)
```

```
BBiitt
```

Aus `""` wird `"BB"`, dann `"BBii"` und schließlich `"BBiitt"`.

Schleife über die Indizes

Brauchst du auch die **Position**, lässt du die Schleife über alle Indizes laufen. `range(len(text))` liefert genau die gültigen Indizes 0, 1, …, `len(text) - 1`.

```
text = "Bit"
for i in range(len(text)):
    print(i, text[i])
```

```
0 B
1 i
2 t
```

> 📌 **Merksatz:** `for zeichen in text:` durchläuft nacheinander alle **Zeichen**. `for i in range(len(text)):` durchläuft alle **Indizes**, das Zeichen dazu ist `text[i]`.

4 Zeichen prüfen und umwandeln

Ist ein Zeichen enthalten?

Mit `in` prüfst du, ob ein Zeichen oder ein Textstück vorkommt. Das Ergebnis ist `True` oder `False`. Besonders praktisch ist `in` mit einer Zeichenkette aller „erlaubten“ Zeichen, etwa `"aeiou"` für die Vokale.

```
print("a" in "Banane")
print("x" in "Banane")
print("nan" in "Banane")
print("B" in "banane")

buchstabe = "e"
if buchstabe in "aeiou":
    print(buchstabe, "ist ein Vokal.")
```

```
True
False
True
False
e ist ein Vokal.
```

Groß- und Kleinschreibung

Für Python sind `"Ja"`, `"ja"` und `"JA"` drei verschiedene Zeichenketten. Ist die Schreibweise egal, wandelt man alles in Klein- oder Großbuchstaben um.

```python
text = "Hallo Welt"
print(text.upper())
print(text.lower())
print(text)
```

```
HALLO WELT
hallo welt
Hallo Welt
```

`text` selbst bleibt **unverändert**: `.upper()` und `.lower()` erzeugen eine neue Zeichenkette. Neu ist die Schreibweise mit dem Punkt **hinter** dem Text. Solche Funktionen heißen **Methoden**.

> 📌 **Definition Methode:** Eine **Methode** ist eine Funktion, die zu einem Wert gehört. Man ruft sie mit einem Punkt hinter dem Wert auf: `text.upper()`. Die Klammern dürfen leer sein, aber nicht fehlen.

⚠️ Stolperstein: `text.lower()` allein ändert nichts

🔮 **Vorhersage:** Was gibt die Zelle aus? Wie musst du Zeile 2 ändern, damit `hallo` herauskommt?

```python
text = "Hallo"
text.lower()
print(text)
```

```
Hallo
```

Buchstabe oder Ziffer?

- `text.isalpha()` ist `True`, wenn **alle** Zeichen Buchstaben sind (auch Umlaute und ß),
- `text.isdigit()` ist `True`, wenn **alle** Zeichen Ziffern sind.

```python
print("Hallo".isalpha())
print("Hallo Welt".isalpha())
print("2026".isdigit())
print("-5".isdigit())
print("3.5".isdigit())
```

```
True
False
True
False
False
```

`"Hallo Welt"` enthält ein Leerzeichen, und das ist kein Buchstabe. Bei `"-5"` und `"3.5"` sind Minuszeichen und Punkt keine Ziffern. Mit `.isdigit()` kann ein Programm eine Eingabe prüfen, bevor es sie mit `int(...)` umwandelt, und so den `ValueError` vermeiden.

> 📌 **Merksatz:** `x in text` prüft, ob `x` enthalten ist. `.upper()` und `.lower()` liefern eine **neue** Zeichenkette, darum `text = text.lower()`. `.isalpha()` und `.isdigit()` prüfen auf Buchstaben bzw. Ziffern.

📝 Übungen: Schritt für Schritt selbstständig

Die Übungen werden schrittweise schwieriger. Vorgemacht hast du schon zwei Muster gesehen: **Zeichen zählen** (`"Anna"`) und **einen neuen Text aufbauen** (`"BBiitt"`). Bei Übung 2 füllst du Lücken `___` aus, bei Übung 3 steht nur noch das Gerüst als Kommentar, bei Übung 4 nur die Kopfzeile und die Tests, die Zusatzaufgabe schreibst du allein.

📝 Übung 2 – Zeichen-Steckbrief (Lücken füllen)

Schreibe ein Programm, das ein Wort einliest und ausgibt: die Länge, das erste und das letzte Zeichen und das Wort rückwärts. So könnte es aussehen:

```
Wort eingeben: Python
Länge: 6
Erstes Zeichen: P
Letztes Zeichen: n
Rückwärts: nohtyP
```

> 💡 **Tipp:** Für das Rückwärts-Wort baust du in einer Schleife einen neuen Text auf wie bei `"BBiitt"`. Der Trick: Hänge jedes Zeichen **vorne** an, also `zeichen + rueckwaerts` statt `rueckwaerts + zeichen`.

Lückencode

```
wort = input("Wort eingeben: ")
print("Länge:", ___)
print("Erstes Zeichen:", wort[___])
print("Letztes Zeichen:", wort[___])

rueckwaerts = ""
for zeichen in wort:
    rueckwaerts = ___
print("Rückwärts:", rueckwaerts)
```

📝 Übung 3 – Vokale zählen (mit Gerüst)

Schreibe ein Programm, das einen Satz einliest und mit einer Schleife zählt, wie viele Vokale (a, e, i, o, u) darin vorkommen. Auch großgeschriebene Vokale sollen mitgezählt werden. Kontrolle: „Informatik macht Spaß“ hat 6 Vokale.

> 💡 **Tipp:** Kombiniere die Schleife über alle Zeichen, die Zählvariable aus dem Beispiel `"Anna"` und die Prüfung `zeichen in "aeiou"`. Gegen das Problem mit den Großbuchstaben hilft `.lower()`.

```
satz = input("Satz eingeben: ")
# Zählvariable auf 0 setzen
# für jedes Zeichen im Satz (in Kleinbuchstaben):
#     wenn das Zeichen in "aeiou" vorkommt:
#         Zählvariable um 1 erhöhen
# nach der Schleife: Anzahl ausgeben
```

```
Satz eingeben: Informatik macht Spaß
```

📝 Übung 4 – Passwort prüfen (Kopfzeile und Tests vorgegeben)

Ein Passwort gilt hier als sicher, wenn es **mindestens 8 Zeichen** lang ist und **mindestens eine Ziffer** enthält. Schreibe eine Funktion `ist_sicher(passwort)`, die `True` oder `False` zurückgibt. Kopfzeile und Testaufrufe stehen schon da.

> 💡 **Tipp:** Zähle die Ziffern wie die Vokale in Übung 3, prüfe aber mit `zeichen.isdigit()`. Verknüpfe am Ende beide Bedingungen mit `and`.

```python
def ist_sicher(passwort):
    pass   # hier deine Programmzeilen

print(ist_sicher("abc1"), "erwartet: False")          # zu kurz
print(ist_sicher("sommerferien"), "erwartet: False")  # keine Ziffer
print(ist_sicher("sommer2026"), "erwartet: True")
```

```
None erwartet: False
None erwartet: False
None erwartet: True
```

🚀 Zusatzaufgabe – Palindrom (selbstständig)

Ein **Palindrom** ist ein Wort, das vorwärts und rückwärts gelesen gleich ist, z. B. „Anna“ oder „Rentner“. Schreibe eine Funktion `ist_palindrom(wort)`, die `True` oder `False` zurückgibt. Groß- und Kleinschreibung soll keine Rolle spielen. Teste mit „Anna“, „Rentner“ und „Informatik“.

➕ Zusatzwissen: Ausschnitte (Slicing)

Für die Aufgaben brauchst du das nicht. Mit `text[start:ende]` schneidest du einen **Ausschnitt** heraus. Das Zeichen am Index `start` gehört dazu, das am Index `ende` **nicht** mehr. `text[::-1]` durchläuft den Text rückwärts.

```python
text = "Informatik"
print(text[0:4])
print(text[2:6])
print(text[::-1])
```

```
Info
form
kitamrofnI
```

📌 Das Wichtigste in Kürze

- Eine **Zeichenkette** ist eine **Sequenz von Zeichen**. Jedes Zeichen hat einen **Index**, gezählt ab 0.
- `text[i]` liefert das Zeichen am Index `i`; `text[-1]` ist das letzte Zeichen.
- `len(text)` liefert die Länge. Gültige Indizes reichen von 0 bis `len(text) - 1`, sonst gibt es einen **IndexError**.
- Mit `+` verkettet man Zeichenketten. Ein **f-String** `f"... {wert} ..."` setzt Werte bequem in einen Text ein.
- `for zeichen in text:` durchläuft alle Zeichen, `for i in range(len(text)):` alle Indizes.
- `x in text` prüft, ob `x` enthalten ist. `.upper()` und `.lower()` liefern eine neue Zeichenkette; `.isalpha()` und `.isdigit()` prüfen auf Buchstaben bzw. Ziffern.

Zum Weiterüben: `DS10_Weitere_Uebungen.ipynb`.