DS 10 – Weitere Übungen

Zeichenketten: Texte Zeichen für Zeichen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Die Aufgaben stehen auch im Skript im Abschnitt „Weitere Übungen“; die Lösungen im Lösungsheft.

Aufgabe 1: Index und Zeichen ●○○

Gegeben ist `wort = "Freibad"`. Bestimme **ohne Computer** die Werte der Ausdrücke. Prüfe danach mit `print(...)` am Computer.

`wort[0]`	`wort[3]`	`wort[-1]`	`len(wort)`	`wort[len(wort) - 2]`

Gib außerdem an, an welchem Index das Zeichen `b` steht.

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

`wort[0]`	`wort[3]`	`wort[-1]`	`len(wort)`	`wort[len(wort) - 2]`
`F`	`i`	`d`	`7`	`a`

Lösung

```
wort = "Freibad"
print(wort[0], wort[3], wort[-1])
print(len(wort))
print(wort[len(wort) - 2])
```

```
F i d
7
a
```

Lösung

Ausgabe:

```
F i d
7
a
```

Lösung

Schreibe dir die Indizes über die Buchstaben: F hat den Index 0, r den Index 1, e den Index 2, i den Index 3, **b den Index 4**, a den Index 5 und d den Index 6. `len(wort) - 2` ist 7 − 2 = 5, also das vorletzte Zeichen.

Aufgabe 2: Ausgabe vorhersagen ●○○

Gib **ohne Computer** an, was das Programm ausgibt. Überprüfe danach am Computer.

```python
text = "Klasse 10a"
print(len(text))
print(text[7] + text[8])
print("s" in text)
print("S" in text)
print(text.lower())
```

```
10
10
True
False
klasse 10a
```

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

```python
text = "Klasse 10a"
print(len(text))
print(text[7] + text[8])
print("s" in text)
print("S" in text)
print(text.lower())
```

```
10
10
True
False
klasse 10a
```

Lösung

Ausgabe:

```
10
10
True
False
klasse 10a
```

Lösung

- `"Klasse 10a"` hat 10 Zeichen, das Leerzeichen zählt mit.
- Am Index 7 steht `1`, am Index 8 steht `0`. Das sind **Zeichen**, keine Zahlen. `+` hängt sie aneinander, das ergibt den Text `10` (und nicht die Zahl 1).
- Ein kleines `s` kommt vor, ein großes `S` nicht.
- `.lower()` macht aus `K` ein `k`. Ziffern und Leerzeichen bleiben, wie sie sind.

Aufgabe 3: Initialen ●○○

Implementiere eine Funktion `initialen(vorname, nachname)`, die die Initialen mit Punkten zurückgibt, zum Beispiel für „Mia“ und „Becker“ die Zeichenkette `M. B.`. Teste die Funktion mit zwei verschiedenen Namen.

Lösung

✅ Lösung

Lösung

```
def initialen(vorname, nachname):
    return vorname[0] + ". " + nachname[0] + "."

print(initialen("Mia", "Becker"))
print(initialen("Jonas", "Schmitt"))
```

```
M. B.
J. S.
```

Lösung

Ausgabe:

```
M. B.
J. S.
```

Lösung

`vorname[0]` ist der erste Buchstabe des Vornamens. Mit `+` werden vier Zeichenketten verkettet: der erste Buchstabe, `". "` (Punkt und Leerzeichen), der erste Buchstabe des Nachnamens und `"."`. Typischer Fehler: `vorname[1]`. Das ist schon der **zweite** Buchstabe.

Aufgabe 4: Buchstaben zählen ●○○

Implementiere eine Funktion `zaehle(text, zeichen)`, die zurückgibt, wie oft `zeichen` in `text` vorkommt. Teste sie mit diesen Aufrufen:

Aufruf	erwartet	tatsächlich
`zaehle("Banane", "a")`	`2`	
`zaehle("Mississippi", "s")`	`4`	
`zaehle("Hallo", "x")`	`0`	

Lösung

✅ Lösung

Lösung

```
def zaehle(text, zeichen):
    anzahl = 0
    for z in text:
        if z == zeichen:
            anzahl = anzahl + 1
    return anzahl

print(zaehle("Banane", "a"))
print(zaehle("Mississippi", "s"))
print(zaehle("Hallo", "x"))
```

```
2
4
0
```

Lösung

Ausgabe:

```
2
4
0
```

Lösung

Das ist das Muster aus dem Beispiel `"Anna"`, verpackt in eine Funktion. Die Schleifenvariable heißt hier `z`, weil der Name `zeichen` schon für den Parameter vergeben ist. Das `return` steht nach der Schleife. Der Test mit `"x"` prüft den Sonderfall, dass das Zeichen gar nicht vorkommt.

Aufgabe 5: Zwei Fehler finden ●●○

Das Programm soll zählen, wie oft der Buchstabe e im Satz vorkommt, egal ob groß oder klein. Erwartet ist 7. Stattdessen bricht es mit einem Fehler ab.

```
satz = "Eine Elefantenherde"
satz.lower()
anzahl = 0
for i in range(1, len(satz) + 1):
    if satz[i] == "e":
        anzahl = anzahl + 1
print(anzahl)
```

```
Traceback (most recent call last):
  File "<Zelle 6>", line 5, in <module>
IndexError: string index out of range
```

1. Gib an, welche Fehlermeldung erscheint, und erkläre sie.
2. Das Programm enthält noch einen zweiten Fehler. Er würde dafür sorgen, dass die großen E nicht mitgezählt werden. Erkläre ihn.
3. Schreibe das Programm richtig auf.

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

1. Es erscheint ein `IndexError`:

Lösung

```
satz = "Eine Elefantenherde"
satz.lower()
anzahl = 0
for i in range(1, len(satz) + 1):
    if satz[i] == "e":
        anzahl = anzahl + 1
print(anzahl)
```

```
Traceback (most recent call last):
  File "<Zelle 7>", line 5, in <module>
IndexError: string index out of range
```

Lösung

Fehlermeldung:

```
IndexError: string index out of range
```

Lösung

Der Satz hat 19 Zeichen, gültig sind also die Indizes 0 bis 18. `range(1, len(satz) + 1)` erzeugt aber die Zahlen 1 bis 19. Der Index 19 existiert nicht. Außerdem wird das erste Zeichen (Index 0) übersprungen. Richtig ist `range(len(satz))`.

2. `satz.lower()` erzeugt zwar den Text in Kleinbuchstaben, aber das Ergebnis wird nicht gespeichert. `satz` bleibt unverändert, und die beiden großen E würden nicht gezählt. Richtig ist `satz = satz.lower()`.
3. Die verbesserte Fassung:

Lösung

```
satz = "Eine Elefantenherde"
satz = satz.lower()
anzahl = 0
for i in range(len(satz)):
    if satz[i] == "e":
        anzahl = anzahl + 1
print(anzahl)
```

```
7
```

Lösung

Ausgabe:

```
7
```

Lösung

Noch einfacher ist die Schleife über die Zeichen: `for zeichen in satz:` mit `if zeichen == "e":`. Dann kann es gar keinen IndexError geben.

Aufgabe 6: Ohne Leerzeichen ●●○

Implementiere eine Funktion `ohne_leerzeichen(text)`, die eine neue Zeichenkette zurückgibt, in der alle Leerzeichen fehlen. Teste mit `"Hallo Welt"` (erwartet: `HalloWelt`) und mit `"0151 234 567"` (erwartet: `0151234567`).

> **Tipp:** Baue wie im Abschnitt *Einen neuen Text aufbauen* einen neuen Text auf. Beginne mit `""` und hänge nur die Zeichen an, die **kein** Leerzeichen sind. Ein Leerzeichen schreibst du als `" "`.

Lösung

✅ **Lösung**

Lösung

```
def ohne_leerzeichen(text):
    neu = ""
    for zeichen in text:
        if zeichen != " ":
            neu = neu + zeichen
    return neu

print(ohne_leerzeichen("Hallo Welt"))
print(ohne_leerzeichen("0151 234 567"))
```

```
HalloWelt
0151234567
```

Lösung

Ausgabe:

```
HalloWelt
0151234567
```

Lösung

Die Schleife geht jedes Zeichen durch. Nur wenn es ungleich `" "` ist, wird es an `neu` angehängt. Leerzeichen werden einfach übersprungen. Typischer Fehler: `neu = ""` in die Schleife schreiben. Dann wird der neue Text in jedem Durchlauf wieder gelöscht, und am Ende steht nur das letzte Zeichen darin.

Aufgabe 7: Vokale tauschen ●●○

Ein bekanntes Sprachspiel: Man ersetzt in einem Satz alle Vokale durch denselben Vokal. Aus „Wir sehen uns morgen“ wird dann „Wir sihin ins mirgin“. Implementiere eine Funktion `nur_i(text)`, die eine neue Zeichenkette zurückgibt, in der jeder Vokal (a, e, i, o, u, auch großgeschrieben) durch ein kleines `i` ersetzt ist. Alle anderen Zeichen bleiben unverändert. Teste mit `"Wir sehen uns morgen"` und mit `"Otto"`.

Lösung

✅ **Lösung**

Lösung

```python
def nur_i(text):
    neu = ""
    for zeichen in text:
        if zeichen in "aeiouAEIOU":
            neu = neu + "i"
        else:
            neu = neu + zeichen
    return neu

print(nur_i("Wir sehen uns morgen"))
print(nur_i("Otto"))
```

```
Wir sihin ins mirgin
itti
```

Lösung

Ausgabe:

```
Wir sihin ins mirgin
itti
```

Lösung

Für jedes Zeichen gibt es zwei Fälle: Ist es ein Vokal, wird `"i"` angehängt, sonst das Zeichen selbst. Die Prüfung `zeichen in "aeiouAEIOU"` erledigt die Groß- und Kleinbuchstaben auf einmal. Der zweite Test zeigt, dass auch ein großer Vokal am Anfang ersetzt wird. Wer möchte, macht aus dem `"i"` einen zweiten Parameter `vokal` und probiert alle Vokale durch.

Aufgabe 8: Handynummer verbergen ●●●

Auf Rechnungen wird eine Handynummer oft nur teilweise gezeigt, zum Beispiel `*********678`. Implementiere eine Funktion `verberge(nummer)`, die eine neue Zeichenkette zurückgibt. Darin ist jedes Zeichen durch `*` ersetzt – außer den **letzten drei** Zeichen. Teste mit `"015112345678"` (erwartet: `*********678`) und `"12345"` (erwartet: `**345`).

> **Tipp:** Hier brauchst du die **Position** jedes Zeichens. Lass die Schleife deshalb über die Indizes laufen: `for i in range(len(nummer)):`. Überlege: Ab welchem Index beginnen die letzten drei Zeichen? Bei 12 Zeichen sind das die Indizes 9, 10 und 11.

Lösung

✅ Lösung

Lösung

```python
def verberge(nummer):
    neu = ""
    for i in range(len(nummer)):
        if i < len(nummer) - 3:
            neu = neu + "*"
        else:
            neu = neu + nummer[i]
    return neu

print(verberge("015112345678"))
print(verberge("12345"))
```

```
*********678
**345
```

Lösung

Ausgabe:

```
*********678
**345
```

Lösung

- **Zeile 3:** `i` durchläuft alle Indizes von 0 bis `len(nummer) - 1`.
- **Zeile 4:** Die letzten drei Zeichen beginnen beim Index `len(nummer) - 3`. Bei 12 Zeichen ist das der Index 9. Alle kleineren Indizes werden durch `*` ersetzt.
- **Zeile 7:** Die letzten drei Zeichen werden mit `nummer[i]` übernommen.

Weil die Grenze mit `len(nummer)` berechnet wird, funktioniert die Funktion für Nummern jeder Länge. Ein häufiger Fehler ist `i < 9`: Das stimmt nur für Nummern mit genau 12 Zeichen.