DS 12 – Weitere Übungen

Algorithmen auf Listen: Summe, Minimum und Suche

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: Maximum von Hand ●○○

Das Programm sucht das Maximum der Liste `[4, 9, 2, 9, 6]`:

```
zahlen = [4, 9, 2, 9, 6]
maximum = zahlen[0]
for zahl in zahlen:
    if zahl > maximum:
        maximum = zahl
print("Maximum:", maximum)
```

```
Maximum: 9
```

Fülle die Tabelle aus: Ist das Element größer als das bisherige Maximum? Welchen Wert hat `maximum` nach diesem Durchlauf? Gib dann an, wie oft `maximum` insgesamt geändert wird.

Durchlauf	`zahl`	`zahl > maximum?`	`maximum` danach
vor der Schleife	–	–	`4`
1	`4`		
2	`9`		
3	`2`		
4	`9`		
5	`6`		

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

Durchlauf	`zahl`	`zahl > maximum?`	`maximum` danach
vor der Schleife	–	–	`4`
1	`4`	nein	`4`
2	`9`	ja	`9`
3	`2`	nein	`9`
4	`9`	nein	`9`
5	`6`	nein	`9`

Lösung

```
zahlen = [4, 9, 2, 9, 6]
maximum = zahlen[0]
for zahl in zahlen:
    if zahl > maximum:
        maximum = zahl
print("Maximum:", maximum)
```

```
Maximum: 9
```

Lösung

Ausgabe:

```
Maximum: 9
```

Lösung

`maximum` wird nur **einmal** geändert, im zweiten Durchlauf. Beim zweiten 9er ist `9 > 9` falsch, denn „größer“ heißt „echt größer“.

Aufgabe 2: Summe und Zählen ●○○

In der Liste stehen die Punkte, die ein Team in sechs Spielen geholt hat. Gib **ohne Computer** an, was das Programm ausgibt, und beschreibe in einem Satz, was `anzahl` zählt.

```
punkte = [3, 0, 1, 3, 0, 3]
summe = 0
anzahl = 0
for p in punkte:
    summe = summe + p
    if p == 0:
        anzahl = anzahl + 1
print(summe, anzahl)
```

```
10 2
```

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

```
punkte = [3, 0, 1, 3, 0, 3]
summe = 0
anzahl = 0
for p in punkte:
    summe = summe + p
    if p == 0:
        anzahl = anzahl + 1
print(summe, anzahl)
```

```
10 2
```

Lösung

Ausgabe:

```
10 2
```

Lösung

Die Summe ist 3 + 0 + 1 + 3 + 0 + 3 = 10. `anzahl` zählt, wie viele Spiele **ohne Punkte** (also verloren) waren: zwei. In derselben Schleife laufen zwei Algorithmen gleichzeitig. Die Summe wird bei **jedem** Element erhöht,

die Zählvariable nur, wenn die Bedingung erfüllt ist.

Aufgabe 3: Playlist ●○○

In einer Liste steht die Länge jedes Songs einer Playlist in Minuten: `songs = [3, 4, 2, 5, 3]`.

1. Implementiere eine Funktion `gesamtdauer(liste)`, die die Summe aller Elemente zurückgibt.
2. Implementiere eine Funktion `mittlere_dauer(liste)`, die die durchschnittliche Länge zurückgibt. Sie soll `gesamtdauer` aufrufen.
3. Teste beide Funktionen mit der Playlist. Erwartet sind 17 Minuten und 3,4 Minuten.

Lösung

✅ **Lösung**

Lösung

```python
def gesamtdauer(liste):
    summe = 0
    for dauer in liste:
        summe = summe + dauer
    return summe

def mittlere_dauer(liste):
    return gesamtdauer(liste) / len(liste)

songs = [3, 4, 2, 5, 3]
print(gesamtdauer(songs), "Minuten")
print(mittlere_dauer(songs), "Minuten")
```

```
17 Minuten
3.4 Minuten
```

Lösung

Ausgabe:

```
17 Minuten
3.4 Minuten
```

Lösung

`gesamtdauer` ist die Summenbildung: Hilfsvariable mit 0 initialisieren, jedes Element addieren, nach der Schleife zurückgeben. `mittlere_dauer` teilt die Summe durch die Anzahl der Songs. So wird die Summe nur an einer Stelle programmiert.

Aufgabe 4: Kälteste Nacht ●○○

In einer Woche wurden nachts diese Temperaturen gemessen (in °C): `nachts = [-3, 2, -7, 0, 4, -1, 1]`.

Implementiere eine Funktion `kaelteste(temperaturen)`, die die niedrigste Temperatur zurückgibt. Verwende nicht `min()`. Teste mit der Liste. Erwartet ist −7.

Lösung

✅ **Lösung**

Lösung

```python
def kaelteste(temperaturen):
    kleinste = temperaturen[0]
    for t in temperaturen:
        if t < kleinste:
            kleinste = t
    return kleinste

nachts = [-3, 2, -7, 0, 4, -1, 1]
print(kaelteste(nachts))
```

```
-7
```

Lösung

Ausgabe:

```
-7
```

Lösung

Das ist der Algorithmus für das Minimum: mit dem ersten Element beginnen und bei jedem kleineren Element aktualisieren. Wer mit `kleinste = 0` beginnt, bekommt hier zufällig das richtige Ergebnis. Bei einer Woche ohne Frost, z. B. `[3, 5, 2]`, käme aber 0 heraus – ein Wert, der gar nicht gemessen wurde.

Aufgabe 5: Freies Schließfach ●●○

In der Liste `faecher` steht, wem welches Schließfach gehört. Ein leeres Fach ist mit der leeren Zeichenkette `""` gekennzeichnet:

```
faecher = ["Ada", "Ben", "", "Mia", ""]
```

1. Implementiere eine Funktion `erstes_freies(faecher)`, die mit einer linearen Suche den **Index** des ersten freien Fachs zurückgibt. Gibt es kein freies Fach, soll sie `-1` zurückgeben.
2. Gib im Hauptprogramm aus, welches Fach frei ist. Die Fächer sind für Menschen ab 1 nummeriert, also ist der Index 2 das Fach Nummer 3. Ist kein Fach frei, soll „Alle Fächer sind belegt.“ erscheinen.
3. Teste auch mit einer Liste ganz ohne freies Fach.

Lösung

✅ **Lösung**

Lösung

```python
def erstes_freies(faecher):
    for i in range(len(faecher)):
        if faecher[i] == "":
            return i
    return -1

faecher = ["Ada", "Ben", "", "Mia", ""]
position = erstes_freies(faecher)
if position == -1:
    print("Alle Fächer sind belegt.")
else:
    print("Fach Nummer", position + 1, "ist frei.")

print(erstes_freies(["Ada", "Ben", "Mia"]))
```

```
Fach Nummer 3 ist frei.
-1
```

Lösung

Ausgabe:

```
Fach Nummer 3 ist frei.
-1
```

Lösung

Die Funktion ist die lineare Suche aus diesem Abschnitt, gesucht wird hier die leere Zeichenkette. Weil `return i` die Funktion sofort beendet, wird das zweite freie Fach (Index 4) nicht mehr angesehen. Das `return -1` steht **nach** der Schleife. Im Hauptprogramm wird das Ergebnis erst geprüft und dann als Fachnummer ausgegeben.

Aufgabe 6: Zwei Fehler im Maximum ●●○

Die Funktion soll die beste Tordifferenz einer Saison zurückgeben. Sie liefert falsche Ergebnisse.

```python
def beste_differenz(differenzen):
    beste = 0
    for d in differenzen:
        if d > beste:
            beste = d
        return beste
```

1. Teste die Funktion mit `[3, -1, 5]` (erwartet: 5) und mit `[-2, -1, -5]` (erwartet: −1). Notiere die Ergebnisse.
2. Erkläre die beiden Fehler und verbessere die Funktion.

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

1. Die fehlerhafte Funktion liefert:

Lösung

```python
def beste_differenz(differenzen):
    beste = 0
    for d in differenzen:
        if d > beste:
            beste = d
        return beste

print(beste_differenz([3, -1, 5]), "erwartet: 5")
print(beste_differenz([-2, -1, -5]), "erwartet: -1")
```

```
3 erwartet: 5
0 erwartet: -1
```

Lösung

Ausgabe:

```
3 erwartet: 5
0 erwartet: -1
```

Lösung

2. **Fehler 1:** `return beste` ist zu weit eingerückt. Es steht in der Schleife und beendet die Funktion schon nach dem ersten Element. Deshalb kommt beim ersten Test 3 heraus. **Fehler 2:** `beste` beginnt mit 0. Sind alle Differenzen negativ, ist keine größer als 0, und die Funktion liefert 0 – einen Wert, der gar nicht in der Liste steht. Richtig ist der Start mit dem ersten Element:

Lösung

```python
def beste_differenz(differenzen):
    beste = differenzen[0]
    for d in differenzen:
        if d > beste:
            beste = d
    return beste

print(beste_differenz([3, -1, 5]), "erwartet: 5")
print(beste_differenz([-2, -1, -5]), "erwartet: -1")
```

```
5 erwartet: 5
-1 erwartet: -1
```

Lösung

Ausgabe:

```
5 erwartet: 5
-1 erwartet: -1
```

Lösung

Der zweite Test war wichtig: Mit lauter positiven Zahlen wäre Fehler 2 nicht aufgefallen.

Aufgabe 7: Frosttage ●●○

Ein Tag mit einer Temperatur **unter** 0 °C heißt Frosttag.

1. Implementiere eine Funktion `frosttage(temperaturen)`, die zählt, wie viele Werte der Liste unter 0 liegen.
2. Teste mit der Liste aus dem Auftrag *Kälteste Nacht* (erwartet: 3) und mit `[0, 1, 2]` (erwartet: 0).

Lösung

✅ **Lösung**

Lösung

```python
def frosttage(temperaturen):
    anzahl = 0
    for t in temperaturen:
        if t < 0:
            anzahl = anzahl + 1
    return anzahl

nachts = [-3, 2, -7, 0, 4, -1, 1]
print(frosttage(nachts), "erwartet: 3")
print(frosttage([0, 1, 2]), "erwartet: 0")
```

```
3 erwartet: 3
0 erwartet: 0
```

Lösung

Ausgabe:

```
3 erwartet: 3
0 erwartet: 0
```

Lösung

Das ist der Zählalgorithmus mit der Bedingung `t < 0`. Der zweite Test prüft die Grenze: 0 °C ist **kein** Frosttag. Mit `t <= 0` käme dort 1 heraus.

Aufgabe 8: Besser als der Durchschnitt ●●●

Implementiere eine Funktion `ueber_durchschnitt(liste)`, die zählt, wie viele Elemente **größer** als der Durchschnitt der Liste sind. Teste mit:

Aufruf	erwartet	tatsächlich
`ueber_durchschnitt([2, 4, 6, 8])`	`2`	
`ueber_durchschnitt([10, 10, 10])`	`0`	
`ueber_durchschnitt([1, 1, 1, 9])`	`1`	

> **Tipp:** Du brauchst **zwei** Schritte nacheinander. Erst berechnest du mit einer Schleife den Durchschnitt. Erst danach kannst du mit einer zweiten Schleife zählen, denn vorher kennst du den Vergleichswert noch nicht.

Lösung

✅ Lösung

Lösung

```python
def ueber_durchschnitt(liste):
    summe = 0
    for wert in liste:
        summe = summe + wert
    mittelwert = summe / len(liste)

    anzahl = 0
    for wert in liste:
        if wert > mittelwert:
            anzahl = anzahl + 1
    return anzahl

print(ueber_durchschnitt([2, 4, 6, 8]))
print(ueber_durchschnitt([10, 10, 10]))
print(ueber_durchschnitt([1, 1, 1, 9]))
```

```
2
0
1
```

Lösung

Ausgabe:

```
2
0
1
```

Lösung

- **Zeilen 2–5:** Summenbildung und Mittelwert. Bei `[2, 4, 6, 8]` ist der Mittelwert 5.
- **Zeilen 7–11:** Zählen mit der Bedingung `wert > mittelwert`. 6 und 8 sind größer als 5, also kommt 2 heraus.

Bei `[10, 10, 10]` ist kein Element **größer** als der Mittelwert 10. Der dritte Test zeigt, dass ein einziger großer Wert den Durchschnitt nach oben zieht (Mittelwert 3). Man kann die Funktion auch mit der Funktion `durchschnitt` aus dem Auftrag *Durchschnitt berechnen* schreiben: `mittelwert = durchschnitt(liste)`.