Listen & Algorithmen

Erwartete Note: ____ **Punkte: ____ / 23 Note: ____**

Aufgabe 1: Listen reparieren . . . . . **/ 4 P.**

Betrachte die Liste `L = [5, 7, 10, 15, 25, 30, 32, 40, 45]`. Verändere die Liste mit vier Zeilen Python-Code, so dass am Schluss die Fünferreihe bis und mit 50 enthalten ist.

Lösung:

```
L.pop(1)
L.insert(3, 20)
L[6] = 35
L.append(50)
```

Bewertung:

- ½P Abzug für Index-Verwirrung
- 1P Abzug für mehr als 4 Befehle
- 1P Abzug für jeden fehlenden / falschen Befehl (ohne Index-Verwirrung)

Aufgabe 2: Schleifen . . . . . **/ 3 P.**

Wandle in der folgenden Funktion die `while`-Schleife in eine möglichst elegante `for`-Schleife um. Das Resultat der Funktion soll das gleiche bleiben.

```
def contains_42(L):
    index = 0
    while index < len(L):
        if L[index] == 42:
            return True
    return False
```

Lösung:

```
def contains_42(L):
    for element in L:
        if element == 42:
            return True
    return False
```

Bewertung:

- 1P Abzug für indirekte `for`-Schleife.
- ½P Abzug für Index-Inkrement / Initialisierung von index

Aufgabe 3: Listen-Algorithmen . . . . . **/ 4 P.**

Schreibe eine Python-Funktion, die alle Elemente in einer Liste zählt, die grösser gleich 100 und kleiner 1000 sind. Die Funktion soll die Liste als Argument erhalten und das Resultat zurückgeben; ein Funktionsaufruf könnte zum Beispiel so aussehen und würde 3 ausgeben:

```
print(zaehlen([155, 25, 999, 1500, 242]))
```

Lösung:

```
def zaehlen(L):
    count = 0
    for n in L:
        if 100 <= n < 1000:
            count = count + 1
    return count
```

Bewertung:

- 1P Abzug für fehlende Funktion
- ½P Abzug für falsche Bedingung (< statt <=)
- kein Abzug für verschachteltes `if`
- kein Abzug für Verwendung von List Comprehensions
- 1P Abzug für Verwendung von globalen Variablen
- 1P Abzug für falsche Platzierung der Zählvariable oder fehlendes `return`

Aufgabe 4: Binäre Suche . . . . . **/ 4 P.**

Die binäre Suche ist viel effizienter als die lineare Suche.

(a) (1 P.) : Illustriere mit Pfeilen in der Darstellung, welche Positionen die binäre Suche bei der Suche nach *Erni* besucht. Wieviele Elemente müssen dazu gelesen werden?

Lösung:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Amato	Bendel-Simmen	Colombo	Conrad	Deiss	Desantis	Engeli	Erni	Furrer	Gentil	Heller	Herzog	Hochstrasser	Hoffmann	Joss	Joumard	Leppert	Lichtensteiger	Nick	Pazeller-Munz	Rothe	Schädlich	Schärer	Schurtenberger	Sigrist	Vonderwahl-Nyffenegger	Widrig	Wüest-Thalmann

(b) (2 P.) : Wieviele Elemente müssen maximal gelesen werden, um in der obigen Liste eine:n Lehrer:in zu finden? Wieviele, wenn alle Menschen der Schweiz (8.7 Millionen) in der Liste verzeichnet wären?

Lösung:
28: 5
8.7M: 24

(c) (1 P.) : Welche Eigenschaft muss eine Liste haben, damit die Binärsuche das richtige Resultat liefert?

Lösung: Die Liste muss sortiert sein.

Aufgabe 5: Umkehrung . . . . . **/ 4 P.**

Schreibe eine Python-Funktion `list_reverse(L)`, die eine neue Liste erstellt und sie mit den Elementen von `L` befüllt, aber in der umgekehrten Reihenfolge. Der Aufruf `list_reverse([7, 42, 21])` soll also eine neue Liste `[21, 42, 7]` zurückgeben. Die übergebene Liste soll nicht verändert werden.

Die Verwendung der eingebauten Python-Funktionen `reverse` und `reversed` ist nicht erlaubt.

Lösung:

```
1  def list_reverse(L):
2      result = []
3      for elem in L:
4          result.insert(0, elem)
5      return result
```

Bewertung:

- 1P Abzug für Veränderung des Arguments
- alle korrekten Implementierungen sind zugelassen (`insert`, `for index in range(len(L)-1, -1, -1)`)

Aufgabe 6: Listen-Algorithmen . . . . . **/ 4 P.**

Schreibe eine Python-Funktion `pop_min(L)`, die das kleinste Element einer Liste `L` entfernt und zurückgibt.

Der folgende Code soll also dazu führen, dass auf der Konsole `13` und `[42, 28, 56, 19]` ausgegeben wird.

```
L = [42, 13, 28, 56, 19]  # Liste zu Beginn
print(pop_min(L))         # Entfernt die 13 und gibt sie aus
print(L)                  # Gibt [42, 28, 56, 19] aus
```

Lösung:

```
def pop_min(L):
    min_index = 0
    minimum = L[min_index]
    for index in range(1,len(L)):
        if L[index] < minimum:
            minimum = L[index]
            min_index = index
    return L.pop(min_index)
```

Bewertung:

- 1P für indirekte `for`-Schleife