1. (4 Punkte) Die folgenden vier Codezeilen verursachen jeweils einen `SyntaxError` bzw. `IndentationError`, bevor das Programm überhaupt startet. Gib jeweils den Fehlertyp an und korrigiere den Code.

```
(a) print("Hello World"
(b) s = 'Montag
(c) def greet():
    print("Hello")
(d) 1x = 'Hallo'
```

Lösung:

```
(a) SyntaxError: schließende Klammer fehlt
    Korrektur: print("Hello World")

(b) SyntaxError: schließendes Hochkomma fehlt
    Korrektur: s = 'Montag'

(c) IndentationError: Einrückung nach dem Doppelpunkt fehlt
    Korrektur:
    def greet():
        print("Hello")

(d) SyntaxError: Variablenname darf nicht mit einer Ziffer beginnen
    Korrektur: x1 = 'Hallo'
```

2. (4 Punkte) Die folgenden fünf Codeausschnitte laufen zunächst an, brechen aber mit einem Laufzeitfehler ab.
(a) IndexError, (b) TypeError, (c) NameError, (d) ValueError, (e) ZeroDivisionError.
Erkläre kurz, warum der Fehler auftritt.

```
(a) a = [1, 2, 3]
    print(a[3])

(b) x = 5 + "10"

(c) print(message)

(d) x = int("abc")

(e) x = 5 / 0
```

Lösung:

```
(a) IndexError: Index 3 existiert nicht (gueltige Indizes: 0-2)
(b) TypeError: int und str koennen nicht mit + verknuepft werden
(c) NameError: Variable message wurde nicht definiert
(d) ValueError: "abc" kann nicht in eine Ganzzahl umgewandelt werden
(e) ZeroDivisionError: Division durch 0
```

3. (5 Punkte) Erkläre die drei Fehlerarten Syntaxfehler, Laufzeitfehler und logischer Fehler. Gib jeweils an, wann der Fehler auftritt (vor dem Start, während der Ausführung oder gar nicht als Absturz erkennbar) und nenne ein kurzes Beispiel.

Lösung:

```
Syntaxfehler:
Das Programm verstößt gegen die Grammatik der Sprache und kann
gar nicht erst gestartet werden.
Beispiel: print("Hallo"       (Klammer fehlt)

Laufzeitfehler:
Das Programm startet, bricht aber während der Ausführung mit
einer Fehlermeldung (Traceback) ab.
Beispiel: x = 5 / 0           (ZeroDivisionError)

Logischer Fehler:
Das Programm startet, läuft bis zum Ende durch und stürzt nicht
ab, liefert aber ein falsches Ergebnis.
Beispiel: durchschnitt = 10 + 20 / 2   (statt (10 + 20) / 2)
```

4. (4 Punkte) Erkläre, was ein Debugger ist und wozu er beim Programmieren genutzt wird. Gehe dabei auf die Begriffe *Einzelschritt*, *Breakpoint* und *Fortfahren* ein.

Lösung: Ein Debugger ist ein Werkzeug der Entwicklungsumgebung, mit dem man ein Programm schrittweise (Anweisung für Anweisung) ausführen und dabei die aktuellen Werte der Variablen beobachten kann. So lassen sich logische Fehler leichter finden, da man genau sieht, wo sich ein Wert anders verändert als erwartet.

Einzelschritt: Führt jeweils nur die nächste Anweisung aus.

Breakpoint: Eine markierte Zeile, an der das Programm automatisch anhält, sodass man nicht jede Zeile einzeln durchgehen muss.

Fortfahren: Lässt das Programm laufen, bis der nächste Breakpoint erreicht wird (oder das Programm endet).

5. (3 Punkte) Gegeben ist die Liste `a = [12, 7, 3, 9, 15, 2]`. Schreibe ein Python-Programm, das die Summe aller Zahlen der Liste berechnet und ausgibt (ohne die eingebaute Funktion `sum` zu verwenden).

Lösung:

```
summe = 0
for x in a:
    summe += x
print(summe)
```

6. (4 Punkte) Gegeben ist die Liste `a = [14, 21, 8, 33, 12, 5, 40]`. Schreibe ein Python-Programm, das zählt, wie viele Zahlen der Liste durch 4 teilbar sind, und die Anzahl ausgibt.

Lösung:

```
zaehl = 0
for x in a:
    if x % 4 == 0:
        zaehl += 1
print(zaehl)
```

7. (4 Punkte) Gegeben ist die Liste `a = [5, 9, 3, 12, 7, 7, 2]`. Schreibe ein Python-Programm, das zählt, wie viele Zahlen der Liste einen größeren Nachfolger haben und die Anzahl ausgibt.

Lösung:

```
a = [5, 9, 3, 12, 7, 7, 2]
zaehl = 0
for i in range(len(a)-1):
    if a[i] < a[i+1]:
        zaehl += 1
print(zaehl)
```

8. (3 Punkte) Gegeben ist die Liste `a = [4, 18, 7, 23, 11, 6]`. Schreibe ein Python-Programm, das mit linearer Suche das kleinste Element der Liste findet und ausgibt (ohne die eingebaute Funktion `min` zu verwenden).

Lösung:

```
a = [4, 18, 7, 23, 11, 6]
best = a[0]
for x in a:
    if x < best:
        best = x
print(best)
```

9. (4 Punkte) Gegeben ist die Liste `a = [8, 15, 22, 6, 19, 30]`. Schreibe ein Python-Programm, das mit linearer Suche das Element findet, das den größten Rest bei der Division durch 5 hat, und dieses ausgibt.

Lösung:

```
a = [8, 15, 22, 6, 19, 30]
best = a[0]
best_val = a[0] % 5
for x in a:
    if x % 5 > best_val:
        best_val = x % 5
        best = x
print(best)
```

10. (4 Punkte) Gegeben ist die Liste `a = [3, 27, 14, 8, 41, 19]`. Schreibe ein Python-Programm, das mit linearer Suche das Element findet, das am nächsten an der Zahl 20 liegt (kleinster Wert von `abs(x - 20)`), und dieses ausgibt.

Lösung:

```
a = [3, 27, 14, 8, 41, 19]
best = a[0]
best_abstand = abs(a[0] - 20)
for x in a:
    if abs(x - 20) < best_abstand:
        best_abstand = abs(x - 20)
        best = x
print(best)
```

11. (4 Punkte) Der folgende Code sucht mit linearer Suche das größte Element einer Liste. Passe den Code so an, dass zusätzlich zum größten Wert auch dessen Index ausgegeben wird.

```
a = [3, 4, 3, 8, 12, 52, 35, 17, 33]
best = a[0]
for x in a:
    if x > best:
        best = x
print(best)
```

Lösung:

```
a = [3, 4, 3, 8, 12, 52, 35, 17, 33]
best = a[0]
best_index = 0
for i in range(len(a)):
    if a[i] > best:
        best = a[i]
        best_index = i
print(best)
print(best_index)
```

12. (4 Punkte) Der folgende Code soll eine Liste mit Bubblesort aufsteigend sortieren. An zwei Stellen (____) fehlt Code. Ergänze die fehlenden Zeilen.

```
def bubble_sort(a):
    getauscht = True
    while getauscht:
        getauscht = ____
        for i in range(len(a)-1):
            if a[i] > a[i+1]:
                a[i], a[i+1] = a[i+1], a[i]
                ____
```

Lösung:

1. Stelle: getauscht = False
   (wird zu Beginn jedes Durchgangs zurückgesetzt)

2. Stelle: getauscht = True
   (wird nach einem Tausch gesetzt, damit noch ein weiterer
   Durchgang stattfindet)