1. (2 Punkte) Gegeben sei eine nicht leere Liste a mit Zahlen. Schreibe ein Python-Programm, das die Liste a durchläuft und zählt, wieviele Zahlen positiv (also größer Null) sind.

```
Beispiel:
a = [15, -8, -14, 42, 8, -42, 16]

Erwartete Ausgabe:
4
```

Lösung:

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

2. (2 Punkte) Gegeben sei eine nicht-leere Liste a mit Zahlen. Schreibe ein Python-Programm, das die Liste a durchläuft und die Summe aller Zahlen ausgibt, die vor einer geraden Zahl stehen.

```
Beispiel:
a = [15, 23, 4, 42, 7, 1, 16]

Erwartete Ausgabe:
28
```

Lösung:

```
summe = 0
for i in range(len(a)-1):
    if a[i+1] % 2 == 0:
        summe += a[i]
print(summe)
```

3. (2 Punkte) Gegeben sei eine nicht-leere Liste a mit mindestens 3 Zahlen. Schreibe ein Python-Programm, das die Liste a durchläuft und die eine neue Liste bildet mit den Summen aus der Zahl und ihren beiden Nachfolgern. Für Zahlen, die keine zwei Nachfolger haben, wird keine Summe gebildet.

```
Beispiel:
a = [1, 2, 3, 4, 5]

Erwartete Ausgabe:
[6, 9, 12]
```

Lösung:

```
b = []
for i in range(len(a)-2):
    b.append(a[i]+a[i+1]+a[i+2])

print(b)
```

4. (3 Punkte) Beschreibe in Worten, welchen Wert die Funktion *doit* ermittelt. Was wird im angegebenen Beispiel ausgegeben?

```
def doit(a):
    best = None
    best_val = -float('inf')
    for i in range(len(a)-1):
        val = a[i+1]
        if val > best_val:
            best_val = val
            best = i
    return best

a = [50,32,66,94,42,14,77,89]
print(doit(a))
```

Lösung:

Die Funktion doit führt eine lineare Suche auf der Liste a durch. Sie ermittelt den Index der Zahl, die den größten Nachfolger hat. Die Ausgabe ist: 2

5. (3 Punkte) Um mit der lineare Suche das größte Element in einer Liste a zu finden, wird die folgende Funktion doit implementiert. Gib eine Liste a an, bei der ein falscher Wert zurückgegeben wird. Gib auch diesen falschen Wert an.

```
def doit(a):
    best = 0
    for i in range(len(a)):
        if a[i] > best:
            best = a[i]
    return best
```

Lösung:

```
a = [-4,-2,-9]
Es wird 0 zurückgegeben.
```

6. (3 Punkte) Gegeben sei eine nicht leere Liste a mit Zahlen. Schreibe ein Python-Programm, das mittels linearer Suche den Index des größten Elements ausgibt. Wenn die größte Zahl mehrfach vorkommt, soll der Index des ersten Vorkommens ausgegeben werden.

```
Beispiel:
a = [15, 23, 4, 42, 8, 42, 16]

Erwartete Ausgabe:
3
```

Lösung:

```
best = None
best_val = -float('inf')
for i in range(len(a)):
    if a[i] > best_val:
        best_val = a[i]
        best = i
print(best)
```

7. (3 Punkte) Gegeben sei eine nicht leere Liste a mit Zahlen. Schreibe ein Python-Programm, das mittels linearer Suche den Index des kleinsten Elements ausgibt. Wenn die kleinste Zahl mehrfach vorkommt, soll der Index des letzten Vorkommens ausgegeben werden.

```
Beispiel:
a = [15, 8, 14, 42, 8, 42, 16]

Erwartete Ausgabe:
4
```

Lösung:

```
best = None
best_val = float('inf')
for i in range(len(a)):
    if a[i] <= best_val:
        best_val = a[i]
        best = i
print(best)
```

8. (3 Punkte) Gegeben sei eine Liste a mit mindestens 3 Zahlen. Schreibe ein Python-Programm, das mittels linearer Suche den Index einer Zahl angibt, bei der die Summe aus linkem und rechtem Nachbarn maximal ist. Zahlen mit nur einem Nachbarn werden nicht berücksichtigt.

```
Beispiel:
a = [15, 8, 14, 42, 8, 42, 16]

Erwartete Ausgabe:
4
```

Lösung:

```
best = None
best_val = -float('inf')
for i in range(1,len(a)-1):
    val = a[i-1] + a[i+1]
    if val > best_val:
        best_val = val
        best = i
print(best)
```