Übungsaufgaben: Funktionen & Scope

Aufgaben

1. **Einfache Begrüßung:** Schreibe eine Funktion namens `sage_hallo`, die keine Parameter entgegennimmt und einfach "Hallo Welt!" auf der Konsole ausgibt.
2. **Quadrat ausgeben:** Schreibe eine Funktion `drucke_quadrat`, die eine Zahl als Parameter entgegennimmt und das Quadrat dieser Zahl auf der Konsole ausgibt.
3. **Personalisierte Begrüßung:** Erstelle eine Funktion `begruesse_person`, die einen Namen als Parameter akzeptiert und eine persönliche Begrüßung ausgibt, z.B. "Hallo, Anna!".
4. **Rechteckfläche ausgeben:** Schreibe eine Funktion `drucke_flaeche`, die zwei Parameter (`breite`, `hoehe`) entgegennimmt und die daraus berechnete Fläche direkt auf der Konsole ausgibt.
5. **Summe zurückgeben:** Erstelle eine Funktion `addiere`, die zwei Zahlen als Parameter erhält und deren Summe mit `return` zurückgibt.
6. **Quadrat zurückgeben:** Modifiziere die Funktion aus Aufgabe 2 so, dass sie das Quadrat der Zahl nicht ausgibt, sondern mit `return` zurückliefert.
7. **Satz erstellen:** Schreibe eine Funktion `erstelle_vorstellung`, die einen Namen und ein Alter als Parameter bekommt und einen vollständigen Satz als String zurückgibt (z.B. "Max ist 30 Jahre alt.").
8. **Volljährigkeit prüfen:** Erstelle eine Funktion `ist_volljaehrig`, die ein Alter entgegennimmt und `true` zurückgibt, wenn das Alter 18 oder größer ist, ansonsten `false`.
9. **Funktionen kombinieren:** Nutze die Funktion `addiere` aus Aufgabe 5. Schreibe Code, der die Funktion zweimal mit unterschiedlichen Werten aufruft, die beiden Ergebnisse in Variablen speichert und dann die Summe dieser beiden Ergebnisse ausgibt.
10. **Listen-Elemente ausgeben:** Schreibe eine Funktion `drucke_einkaufsliste`, die eine Liste von Strings (z.B. `["Milch", "Brot"]`) als Parameter entgegennimmt und jeden Eintrag der Liste in einer neuen Zeile ausgibt.
11. **Listensumme berechnen:** Erstelle eine Funktion `berechne_listensumme`, die eine Liste von Zahlen als Parameter erhält und die Summe aller Zahlen in der Liste zurückgibt.
12. **Kleinste Zahl finden:** Schreibe eine Funktion `finde_minimum`, die eine Liste von Zahlen entgegennimmt und die kleinste Zahl aus der Liste zurückgibt (ohne die eingebaute `min()`-Funktion zu verwenden).
13. **Positive Zahlen filtern:** Erstelle eine Funktion `filtere_positive_zahlen`, die eine Liste mit positiven und negativen Zahlen erhält und eine **neue** Liste zurückgibt, die nur die positiven Zahlen enthält.
14. **Liste verändern (Pass-by-Reference):** Schreibe eine Funktion `verdopple_werte`, die eine Liste von Zahlen als Parameter bekommt und jeden Wert in der Liste direkt verdoppelt. Die Funktion soll **keinen** `return`-Wert haben. Überprüfe außerhalb der Funktion, ob sich die ursprüngliche Liste verändert hat.
15. **Scope-Fehler provozieren:** Schreibe eine Funktion, in der du eine lokale Variable (z.B. `geheimnis`) definierst. Versuche nach dem Aufruf der Funktion, auf diese Variable zuzugreifen. Erkläre in einem Kommentar im Code, warum dies zu einem Fehler führt.
16. **Shadowing demonstrieren:** Definiere eine globale Variable `x = "global"`. Schreibe eine Funktion, die ebenfalls eine Variable `x` (z.B. `x = "lokal"`) definiert und ausgibt. Gib die globale Variable `x` nach dem Funktionsaufruf erneut aus, um zu zeigen, dass sie unverändert ist.
17. **Taschenrechner-Funktion:** Erstelle eine Funktion `rechner`, die drei Parameter akzeptiert: zwei Zahlen und einen String für die Operation (`"+"`, `"-"`, `"*"` oder `"/"`). Die Funktion soll die entsprechende Berechnung durchführen und das Ergebnis zurückgeben.
18. **Vokalzähler-Funktion:** Schreibe eine Funktion `zaehle_vokale`, die einen Text entgegennimmt und die Anzahl der Vokale (a, e, i, o, u, unabhängig von Groß-/Kleinschreibung) in diesem Text zurückgibt.
19. **Primzahl-Funktion:** Erstelle eine Funktion `ist_primzahl`, die eine Zahl als Parameter erhält und `true` zurückgibt, falls es sich um eine Primzahl handelt, ansonsten `false`.
20. **Programm-Refactoring:** Nimm die Logik des "Versandkosten-Rechners" aus den vorherigen Übungen. Deine Aufgabe ist es, den Code in zwei Funktionen zu strukturieren:
    - `lies_gewicht()`: Fragt den Benutzer nach dem Gewicht und gibt es zurück.
    - `berechne_versandkosten(gewicht)`: Nimmt das Gewicht entgegen und gibt die Kosten zurück. Der Hauptteil des Programms soll nur noch diese beiden Funktionen aufrufen.

Lösungen

🐍 Python Lösungen

```python
# 1. Einfache Begrüßung
def sage_hallo():
    print("Hallo Welt!")

# 2. Quadrat ausgeben
def drucke_quadrat(zahl):
    print(zahl * zahl)

# 3. Personalisierte Begrüßung
def begruesse_person(name):
    print(f"Hallo, {name}!")

# 4. Rechteckfläche ausgeben
def drucke_flaeche(breite, hoehe):
    print(f"Die Fläche beträgt: {breite * hoehe}")

# 5. Summe zurückgeben
def addiere(a, b):
    return a + b

# 6. Quadrat zurückgeben
def gib_quadrat(zahl):
    return zahl * zahl

# 7. Satz erstellen
def erstelle_vorstellung(name, alter):
    return f"{name} ist {alter} Jahre alt."

# 8. Volljährigkeit prüfen
def ist_volljaehrig(alter):
    return alter >= 18

# 9. Funktionen kombinieren
ergebnis1 = addiere(5, 10)
ergebnis2 = addiere(3, 7)
gesamtsumme = ergebnis1 + ergebnis2
print(f"Die Gesamtsumme ist: {gesamtsumme}")

# 10. Listen-Elemente ausgeben
def drucke_einkaufsliste(elemente):
    for item in elemente:
        print(item)

# 11. Listensumme berechnen
def berechne_listensumme(zahlen):
    summe = 0
    for zahl in zahlen:
        summe += zahl
    return summe

# 12. Kleinste Zahl finden
def finde_minimum(zahlen):
    minimum = zahlen[0]
    for zahl in zahlen:
        if zahl < minimum:
            minimum = zahl
    return minimum

# 13. Positive Zahlen filtern
def filtere_positive_zahlen(zahlen):
```

```python
    positive_liste = []
    for zahl in zahlen:
        if zahl > 0:
            positive_liste.append(zahl)
    return positive_liste

# 14. Liste verändern
def verdopple_werte(zahlen):
    for i in range(len(zahlen)):
        zahlen[i] *= 2

# 15. Scope-Fehler provozieren
def scope_test():
    geheimnis = "Das ist lokal."
scope_test()
# print(geheimnis) # Fehler: NameError: name 'geheimnis' is not defined
# Die Variable 'geheimnis' existiert nur innerhalb der Funktion 'scope_test'.

# 16. Shadowing demonstrieren
x = "global"
def shadowing_test():
    x = "lokal"
    print(f"In der Funktion ist x: {x}")
shadowing_test()
print(f"Außerhalb ist x: {x}")

# 17. Taschenrechner-Funktion
def rechner(a, b, operation):
    if operation == "+":
        return a + b
    elif operation == "-":
        return a - b
    elif operation == "*":
        return a * b
    elif operation == "/":
        return a / b

# 18. Vokalzähler-Funktion
def zaehle_vokale(text):
    anzahl = 0
    for char in text.lower():
        if char in "aeiou":
            anzahl += 1
    return anzahl

# 19. Primzahl-Funktion
def ist_primzahl(zahl):
    if zahl <= 1:
        return False
    for i in range(2, int(zahl**0.5) + 1):
        if zahl % i == 0:
            return False
    return True

# 20. Programm-Refactoring
def lies_gewicht():
    return float(input("Gewicht des Pakets eingeben: "))
def berechne_versandkosten(gewicht):
    if gewicht <= 2:
        return 5
    elif gewicht <= 5:
        return 8
    else:
        return 12

# Hauptprogramm (simuliert)
# paket_gewicht = lies_gewicht()
```

```
# kosten = berechne_versandkosten(paket_gewicht)
# print(f"Die Kosten betragen: {kosten}€")
```

📜 JavaScript Lösungen

```javascript
// 1. Einfache Begrüßung
function sageHallo() { console.log("Hallo Welt!"); }

// 2. Quadrat ausgeben
function druckeQuadrat(zahl) { console.log(zahl * zahl); }

// 3. Personalisierte Begrüßung
function begruessePerson(name) { console.log(`Hallo, ${name}!`); }

// 4. Rechteckfläche ausgeben
function druckeFlaeche(breite, hoehe) { console.log(`Die Fläche beträgt: ${breite * hoehe}`); }

// 5. Summe zurückgeben
function addiere(a, b) { return a + b; }

// 6. Quadrat zurückgeben
function gibQuadrat(zahl) { return zahl * zahl; }

// 7. Satz erstellen
function erstelleVorstellung(name, alter) { return `${name} ist ${alter} Jahre alt.`; }

// 8. Volljährigkeit prüfen
function istVolljaehrig(alter) { return alter >= 18; }

// 9. Funktionen kombinieren
let ergebnis1 = addiere(5, 10);
let ergebnis2 = addiere(3, 7);
let gesamtsumme = ergebnis1 + ergebnis2;
console.log(`Die Gesamtsumme ist: ${gesamtsumme}`);

// 10. Listen-Elemente ausgeben
function druckeEinkaufsliste(elemente) {
    elemente.forEach(item => console.log(item));
}

// 11. Listensumme berechnen
function berechneListensumme(zahlen) {
    let summe = 0;
    for (const zahl of zahlen) { summe += zahl; }
    return summe;
}

// 12. Kleinste Zahl finden
function findeMinimum(zahlen) {
    let minimum = zahlen[0];
    for (const zahl of zahlen) { if (zahl < minimum) { minimum = zahl; } }
    return minimum;
}

// 13. Positive Zahlen filtern
function filterePositiveZahlen(zahlen) {
    let positiveListe = [];
    for (const zahl of zahlen) { if (zahl > 0) { positiveListe.push(zahl); } }
    return positiveListe;
}

// 14. Liste verändern
function verdoppleWerte(zahlen) {
    for (let i = 0; i < zahlen.length; i++) { zahlen[i] *= 2; }
}

// 15. Scope-Fehler provozieren
function scopeTest() {
```

```
    const geheimnis = "Das ist lokal.";
}
scopeTest();
// console.log(geheimnis); // Fehler: ReferenceError: geheimnis is not defined
// Die Konstante 'geheimnis' existiert nur innerhalb der Funktion 'scopeTest'.

// 16. Shadowing demonstrieren
let x = "global";
function shadowingTest() {
    let x = "lokal";
    console.log(`In der Funktion ist x: ${x}`);
}
shadowingTest();
console.log(`Außerhalb ist x: ${x}`);

// 17. Taschenrechner-Funktion
function rechner(a, b, operation) {
    if (operation === "+") return a + b;
    if (operation === "-") return a - b;
    if (operation === "*") return a * b;
    if (operation === "/") return a / b;
}

// 18. Vokalzähler-Funktion
function zaehleVokale(text) {
    let anzahl = 0;
    const vokale = "aeiou";
    for (const char of text.toLowerCase()) { if (vokale.includes(char)) { anzahl++; } }
    return anzahl;
}

// 19. Primzahl-Funktion
function istPrimzahl(zahl) {
    if (zahl <= 1) return false;
    for (let i = 2; i <= Math.sqrt(zahl); i++) {
        if (zahl % i === 0) return false;
    }
    return true;
}

// 20. Programm-Refactoring
function liesGewicht() {
    // In einer echten Umgebung: return parseFloat(prompt("Gewicht:"));
    return 3.5; // simuliert
}
function berechneVersandkosten(gewicht) {
    if (gewicht <= 2) return 5;
    if (gewicht <= 5) return 8;
    return 12;
}

// Hauptprogramm
// const paketGewicht = liesGewicht();
// const kosten = berechneVersandkosten(paketGewicht);
// console.log(`Die Kosten betragen: ${kosten}€`);
```

☕ Java Lösungen

```java
import java.util.ArrayList;
import java.util.List;

public class UebungenFunktionen {
    // 1. Einfache Begrüßung
    public static void sageHallo() { System.out.println("Hallo Welt!"); }

    // 2. Quadrat ausgeben
    public static void druckeQuadrat(int zahl) { System.out.println(zahl * zahl); }

    // 3. Personalisierte Begrüßung
    public static void begruessePerson(String name) { System.out.println("Hallo, " + name +
"!"); }

    // 4. Rechteckfläche ausgeben
    public static void druckeFlaeche(double breite, double hoehe) { System.out.println("Die
Fläche beträgt: " + (breite * hoehe)); }

    // 5. Summe zurückgeben
    public static int addiere(int a, int b) { return a + b; }

    // 6. Quadrat zurückgeben
    public static int gibQuadrat(int zahl) { return zahl * zahl; }

    // 7. Satz erstellen
    public static String erstelleVorstellung(String name, int alter) { return name + " ist " +
alter + " Jahre alt."; }

    // 8. Volljährigkeit prüfen
    public static boolean istVolljaehrig(int alter) { return alter >= 18; }

    // 10. Listen-Elemente ausgeben
    public static void druckeEinkaufsliste(List<String> elemente) {
        for (String item : elemente) { System.out.println(item); }
    }

    // 11. Listensumme berechnen
    public static int berechneListensumme(List<Integer> zahlen) {
        int summe = 0;
        for (int zahl : zahlen) { summe += zahl; }
        return summe;
    }

    // 12. Kleinste Zahl finden
    public static int findeMinimum(List<Integer> zahlen) {
        int minimum = zahlen.get(0);
        for (int zahl : zahlen) { if (zahl < minimum) { minimum = zahl; } }
        return minimum;
    }

    // 13. Positive Zahlen filtern
    public static List<Integer> filterePositiveZahlen(List<Integer> zahlen) {
        List<Integer> positiveListe = new ArrayList<>();
        for (int zahl : zahlen) { if (zahl > 0) { positiveListe.add(zahl); } }
        return positiveListe;
    }

    // 14. Liste verändern
    public static void verdoppleWerte(List<Integer> zahlen) {
        for (int i = 0; i < zahlen.size(); i++) {
            zahlen.set(i, zahlen.get(i) * 2);
        }
    }
```

```java
// 15. Scope-Fehler provozieren
public static void scopeTest() {
    String geheimnis = "Das ist lokal.";
}

// 16. Shadowing demonstrieren
static String x = "global";
public static void shadowingTest() {
    String x = "lokal";
    System.out.println("In der Methode ist x: " + x);
}

// 17. Taschenrechner-Funktion
public static double rechner(double a, double b, String operation) {
    switch (operation) {
        case "+": return a + b;
        case "-": return a - b;
        case "*": return a * b;
        case "/": return a / b;
        default: return 0.0;
    }
}

// 18. Vokalzähler-Funktion
public static int zaehleVokale(String text) {
    int anzahl = 0;
    String vokale = "aeiou";
    for (char c : text.toLowerCase().toCharArray()) {
        if (vokale.indexOf(c) != -1) { anzahl++; }
    }
    return anzahl;
}

// 19. Primzahl-Funktion
public static boolean istPrimzahl(int zahl) {
    if (zahl <= 1) return false;
    for (int i = 2; i <= Math.sqrt(zahl); i++) {
        if (zahl % i == 0) return false;
    }
    return true;
}

// 20. Programm-Refactoring
public static double liesGewicht() {
    // In einer echten Umgebung: new Scanner(System.in).nextDouble();
    return 3.5; // simuliert
}
public static int berechneVersandkosten(double gewicht) {
    if (gewicht <= 2) return 5;
    if (gewicht <= 5) return 8;
    return 12;
}

public static void main(String[] args) {
    // Hier können die Funktionen aufgerufen werden, z.B.:
    // 9. Funktionen kombinieren
    int ergebnis1 = addiere(5, 10);
    int ergebnis2 = addiere(3, 7);
    int gesamtsumme = ergebnis1 + ergebnis2;
    System.out.println("Die Gesamtsumme ist: " + gesamtsumme);

    // 15. Scope-Fehler provozieren
    scopeTest();
    // System.out.println(geheimnis); // Fehler: cannot find symbol
    // Die Variable 'geheimnis' existiert nur innerhalb der Methode 'scopeTest'.
```

```
        // 16. Shadowing demonstrieren
        shadowingTest();
        System.out.println("Außerhalb ist x: " + x);
    }
}
```