

(Die nachfolgenden Aufgaben wurden mit Hilfe von ChatGPT erstellt, überprüft und ggf. angepasst)

Übungsaufgaben zur 0. Vorlesung: Strings, Zahlen, Rechnungen, Benutzereingabe, print, format, slicing

1. Aufgabenstellung:

Du arbeitest bei einem Paketdienst und musst berechnen, wie viel ein Paket kostet, abhängig vom Gewicht. Der Benutzer gibt das Gewicht des Pakets in Kilogramm ein, und dein Programm berechnet den Preis anhand eines festen Kilopreises.

- Preis pro Kilogramm: **2.99 Euro**

Beispiel-Ablauf:

Wie schwer ist das Paket in Kilogramm? 3.5

Der Preis für den Versand beträgt 10.47 Euro.

2. Aufgabenstellung:

Schreibe ein kleines Programm, das berechnet, wie viel Trinkgeld du in einem Restaurant geben solltest. Der Benutzer gibt den Rechnungsbetrag und den gewünschten Trinkgeld-Prozentsatz ein. Das Programm soll dann das Trinkgeld berechnen und den Gesamtbetrag (Rechnung + Trinkgeld) ausgeben.

Beispiel-Ablauf:

Wie hoch ist die Rechnung in Euro? 45.50

Wie viel Prozent Trinkgeld möchtest du geben? 10

Du solltest 4.55 Euro Trinkgeld geben.

Der Gesamtbetrag ist 50.05 Euro.

Tipps (wären in einer Klausur **nicht** Teil der Aufgabenstellung):

In beiden Aufgaben benötigst Du

- `input()` zum Einlesen von Daten
- `float()` zum Umwandeln von Strings in Zahlen
- eine Rechnung (Multiplikation / Addition / Division)
- `print()` und `format()` zur formatierten Ausgabe

3. Aufgabenstellung:

Du sollst ein kleines Programm schreiben, das aus einer Ticketnummer bestimmte Informationen extrahiert.

Ein Ticketcode ist ein String mit folgendem Aufbau:

JJMMTT-SSS-K

- JJMMTT = Datum des Films (Jahr, Monat, Tag, z. B. 250415 für 15.04.2025)
- SSS = Sitznummer (z. B. 045)
- K = Kategorie des Tickets:
 - N (= Normal)
 - E (= Ermäßigt)
 - L (= Luxus)

Beispiel-Ablauf:

Bitte gib den Ticketcode ein: 250415-123-N

Film am 15.04.2025, Sitz 123, Kategorie: N

Tipps: Hierfür benötigst du „slicing“, also die Möglichkeit mit eckigen Klammern auf bestimmte Elemente des Eingabstrings zuzugreifen.

4. Aufgabenstellung:

Der Benutzer gibt zwei Uhrzeiten im Format hhmm (z. B. 0830 oder 2215) als Strings ein. Dein Programm soll die Zeitdifferenz in Minuten berechnen und anzeigen, wie lange es von der ersten bis zur zweiten Uhrzeit dauert. Jede Uhrzeit ist als String der Länge 4 gegeben, wobei die ersten zwei Zeichen die Stunden und die letzten zwei Zeichen die Minuten sind. Du kannst davon ausgehen, dass die Endzeit immer größer ist als die Startzeit – einen „Überlauf“ bei 24:00 Uhr musst du nicht berücksichtigen.

Beispiel-Ablauf:

Gib die Startzeit ein (hhmm): 0830

Gib die Endzeit ein (hhmm): 1015

Die Zeitdifferenz beträgt 105 Minuten.

Tipps: Wandle (mittels Slicing) zunächst beide Uhrzeiten in Minuten um. Bilde dann die Differenz.

Übungsaufgaben zur 1. Vorlesung: if, for, while, listen, komplexere Abläufe wie Spiele

Aufgabenstellung 3b): Erweiterung von 3. mit if-Abfrage

Ändere Dein Programm aus Aufgabe 3 so, dass nun nicht nur der Code der Kategorie ausgegeben wird, sondern die Worte „Normal“, „Ermäßigt“ oder „Luxus“. Unter der Annahme, dass die Kategorie „Normal“ 10 Euro kostet, die Kategorie „Ermäßigt“ 8 Euro und die Kategorie „Luxus“ 12,50 Euro, gib den Preis für das Ticket mit aus.

Beispiel-Ablauf:

Bitte gib den Ticketcode ein: 250415-123-N

Film am 15.04.2025, Sitz 123, Kategorie: Normal

Der Ticketpreis beträgt 8 Euro.

Aufgabenstellung 4b): Erweiterung von 4. mit if-Abfrage

Ändere Dein Programm aus Aufgabe 4 so, dass die Endzeit kleiner sein kann als die Anfangszeit. In diesem Fall soll dein Programm annehmen, dass es sich um den nächsten Tag handelt.

Beispiel-Ablauf:

Gib die Startzeit ein (hhmm): 2230

Gib die Endzeit ein (hhmm): 0115

Die Zeitdifferenz beträgt 165 Minuten.

5. Aufgabenstellung

Schreibe ein kleines Zahlen-Ratespiel. Der Benutzer soll eine geheime Zahl erraten, die vorher im Code festgelegt ist. Er darf so lange raten, bis er die richtige Zahl eingegeben hat. Nach jedem Versuch bekommt er Feedback, ob seine Zahl zu klein, zu groß oder richtig war.

Beispiel-Ablauf:

Rate die Zahl: 30

Zu klein!

Rate die Zahl: 80

Zu groß!

Rate die Zahl: 42

Richtig! Die geheime Zahl war 42.

Was du tun sollst:

1. Lege am Anfang eine geheime Zahl fest (z. B. geheim = 42)
2. Benutze eine while-Schleife, um den Benutzer so lange raten zu lassen, bis er die richtige Zahl eingibt.
3. Vergleiche mit Hilfe einer if-Abfrage, ob die geratene Zahl größer, kleiner oder gleich der geheimen Zahl ist und gib dem Benutzer die entsprechende Rückmeldung.
4. Lege am Anfang eine boolsche Variable names „noch_nicht_erraten“ an und setze ihren Wert anfangs auf True. Wieso hilft dir dies die while-Schleife zu steuern?

6. Aufgabenstellung:

Schreibe ein Programm, das den Benutzer nach einem Passwort fragt. Er hat maximal 3 Versuche, um das richtige Passwort einzugeben. Wenn das richtige Passwort eingegeben wird, erscheint eine Erfolgsmeldung. Wenn nach 3 Versuchen das Passwort immer noch falsch ist, wird das Programm beendet. Hierfür soll eine while-Schleife verwendet werden.

Beispiel-Ablauf:

Gib das Passwort ein: banane

Falsch! Noch 2 Versuche.

Gib das Passwort ein: apfel

Falsch! Noch 1 Versuch.

Gib das Passwort ein: kirsche

Zugang verweigert.

Oder – bei richtigem Passwort:

Gib das Passwort ein: geheim123

Zugang gewährt!

Was du tun sollst:

1. Lege ein geheimes Passwort fest, z.B. `korrektes_passwort = "geheim123"`
2. Benutze eine Variable zum Zählen der Versuche.
3. Solange das Passwort nicht stimmt und die Anzahl der Versuche unter 3 liegt, frage den Benutzer mit `input()`.
4. Wenn das Passwort stimmt: Erfolgsmeldung
5. Lege geeignete Bedingungen für das Fortführen der while-Schleife fest. Ggf. nutze eine boolsche Variable, z.B. „`passwort_noch_nicht_erraten = True`“. Welche weitere Bedingung wird in der while-Schleife benötigt?

7. Aufgabenstellung (Verwendung von fast allen bisherigen Konzepten)

Schreibe ein Programm, mit dem der Benutzer eine Einkaufsliste erstellt. Er kann nacheinander Produkte eingeben, jeweils mit dem Namen und dem Preis (z. B. Milch:1.29). Solange der Benutzer nicht "fertig" eingibt, soll das Programm weitere Produkte aufnehmen.

Am Ende gibst du aus:

- wie viele Produkte eingegeben wurden,
- wie hoch der Preis für alle Produkte zusammen ist,
- und welche Produkte auf der Liste stehen.

Beispiel-Ablauf:

Produkt eingeben (Name:Preis oder 'fertig'): Milch:1.29

Produkt eingeben (Name:Preis oder 'fertig'): Brot:2.49

Produkt eingeben (Name:Preis oder 'fertig'): Käse:3.99

Produkt eingeben (Name:Preis oder 'fertig'): fertig

Eingekaufte Produkte:

Milch

Brot

Käse

Anzahl: 3

Gesamtkosten: 7.77 Euro

Was du tun sollst:

1. Verwende eine while-Schleife, um wiederholt Eingaben entgegenzunehmen.
2. Zerlege jeden Eingabestring mit `split(":")` in Name und Preis.
3. Speichere die Namen in einer Liste, z. B. `produkte = []`
4. Summiere die Preise auf.
5. Beende die Schleife bei "fertig". Verwende eine for-Schleife, um alle Produkte auszugeben. Danach gibst du die Zusammenfassung der Anzahl und der Gesamtkosten mit `print()` und `format()` aus.