Übung 3 - Schleifen und Funktionen

Programmiervorkurs 2025

1 Schleifen

1.1 Summe

In dieser Aufgabe soll das Berechnen der Summe der Zahlen 1 bis 20 (eingeschlossen) auf unterschiedlichen Wegen implementiert werden.

1.1.1 Gaußsche Summenformel

Um eine Referenz zu haben und mit einem einfachen Beispiel zu beginnen, soll zuerst Berechnung einer Summe mithilfe der Gaußschen Summenformel implementiert werden.

$$1 + 2 + 3 + \ldots + n = \sum_{k=1}^{n} k = \frac{n \cdot (n+1)}{2}$$

```
n = 20
summe = n * (n+1) / 2
print(f"Die Summe der ersten {b} Zahlen ist {summe}")
```

1.1.2 while-Schleife

Da nicht alle Zahlenreihen einfach aufsteigend und damit mit der Gaußschen Summenformel berechenbar sind, soll nun die Summe unter Verwendung einer **while**-Schleife implementiert werden.

```
n = 20
summe = 0
while i <= n:
    summe += i
    i += 1
print(f"Die Summe der ersten {n} Zahlen ist {summe}")
```

1.1.3 for-Schleife

Abschließend soll die Summe mithilfe einer **for**-Schleife umgesetzt werden.

```
n = 20
summe = 0
for i in range(1, n+1):
    summe += i
print(f"Die Summe der ersten {n} Zahlen ist {summe}")
```

1.2 Fakultät

1. Schreibe ein Programm, das den Wert des Ausdrucks $1*2*3*\ldots*15 = 15!$ (Fakultät von 15) berechnet und das Ergebnis auf der Konsole ausgibt.
2. Erweitere dein Programm, sodass es von beliebigen Eingaben in der Konsole die Fakultät berechnet.

```python
n = int(input('Gib eine Zahl an: '))
result = 1
for i in range(n):
    result *= (i+1)

print(f'Die Fakultät von {n} ist {result}')
```

1.3 Listenverarbeitung

Häufig gibt es in Programmen eine Variable, in der das Ergebnis langsam aufgebaut wird. Beispielsweise ist diese für eine Summe zu Anfang 0. Danach wird diese in einer Schleife für jeden Eintrag verändert. Schreibe ein Programm, das die für die Liste `[10, 2, 7, 4, 3, 5, 8]` die folgenden Werte berechnet:

1. Die Summe aller Elemente
2. Das Produkt aller Elemente
3. Der Durchschnitt aller Elemente
4. Eine Liste der Quadrate der Elemente

Was musst du bei der Berechnung des Durchschnitts beachten?
Schaffst du es, die Aufgabe mit nur einer Schleife zu lösen?

> *Bei der Berechnung des Durchschnitts ist zu beachten, dass die Liste auch leer sein kann, was zu einer unerlaubten Division durch Null führen kann.*

```python
liste = [10, 2, 7, 4, 3, 5, 8]

summe = 0
produkt = 1
quadrate = []

for zahl in liste:
    summe = summe + zahl
    produkt = produkt * zahl
    quadrate.append(zahl * zahl)

print(summe)
print(produkt)
print(float(summe) / len(liste) if len(liste) == 0 else "Leere Liste!")
print(quadrate)
```

1.4 Primzahl

In dieser Aufgabe soll ein Programm zur Erkennung von Primzahlen geschrieben werden. Lasse dein Programm dafür eine Zahl mittels `input()` einlesen und überprüfen, ob diese eine Primzahl ist. Teste dein Programm mit verschiedenen Werten.

```python
num = int(input("Gib eine Zahl an: "))
is_prime = True
for i in range(2, num):
```

```
    if num % i == 0:
        is_prime = False
        break
print(f'{num} ist {'' if is_prime else 'k'}eine Primzahl')
```

Effizienter wäre es natürlich, wenn man in der Schleife nur bis $\sqrt{num}$ iteriert.

2 Dateien

In dieser Aufgabe soll der Umgang mit Dateien beigebracht werden.

2.1 Daten schreiben

Info

Bitte vergewissert euch, dass der Pfad korrekt ist und ihr keine existierenden Dateien überschreibt, die euch wichtig sind!

Folgende Daten einer fiktiven Bestellung sollen im Format ”comma-separated values” (kurz CSV)[1] in eine Datei geschrieben werden:

```
bestellungen = [
    ('joern.siegmund@gmail.com', 'Jörn Siegmund', 'Informatik'),
    ('alena_joerg@aol.com', 'Alena Jörg', 'Elektrotechnik'),
    ('gema76@yahoo.com', 'Gerta Volkmar', 'Informatik'),
    ('lia.wendeltreppe@proton.me', 'Lia Wendel', 'Cognitive Science'),
    ('leon.heinrich@gmail.com', 'Leon Heinrich', 'Wirtschaftsinformatik'),
    ('harter-mut1996@gmx.de', 'Christoph Hartmut', 'Maschinenbau')
]
```

Erstellt dazu mithilfe von Python eine neue Datei (mit Dateiendung `.csv`) und schreibt dazu die Werte in die erstellte Datei. Jedes Tuple soll dabei in eine eigene Zeile, wobei die einzelnen Daten des Tuples mit Kommata separiert sein sollen.

Überprüft nach der Ausführung im Texteditor eures Vertrauens, ob die Werte korrekt geschrieben wurden. Macht euch außerdem Gedanken, welche Arten von Eingaben eventuell Probleme mit dem verwendeten Format machen könnten.

```
bestellungen = [
    ('joern.siegmund@gmail.com', 'Jörn Siegmund', 'Informatik'),
    ('alena_joerg@aol.com', 'Alena Jörg', 'Elektrotechnik'),
    ('gema76@yahoo.com', 'Gerta Volkmar', 'Informatik'),
    ('lia.wendeltreppe@proton.me', 'Lia Wendel', 'Cognitive Science'),
    ('leon.heinrich@', 'Leon Heinrich', 'Wirtschaftsinformatik'),
    ('harter-mut1996@gmx.de', 'Christoph Hartmut', 'Maschinenbau')
]

with open('bestellungen.csv', 'w') as f:
    for bestellung in bestellungen:
        mail, name, studiengang = bestellungen[idx]
        f.write(f'{mail},{name},{studiengang}\n')
```

[1] Wikipedia-Artikel zu CSV

2.2 Daten lesen

In der letzten Teilaufgabe habt ihr (hoffentlich erfolgreich) Dateien erstellt und in diese geschrieben. In dieser Aufgabe sollen nun diese Daten wieder eingelesen werden. Öffnet dazu die in der letzten Teilaufgabe erstellte Datei und lest alle Zeilen als Liste von Strings ein. Zählt, wie häufig einzelne Studiengänge angegeben wurden und gibt diese in der Konsole aus.

```python
with open('bestellungen.csv') as f:
    lines = f.readlines()

studiengang = dict()
for line in lines:
    data = line.split(',')
    studiengang[data[2]] = studiengang.get(data[2], 0) + 1

for s in studiengang:
    print(f'{s.replace("\n", "")}: {studiengang[s]}')
```

3 Funktionen

3.1 Einstieg

1. Als Einstieg soll eine Hilfsfunktionen zum Umrechnen von Einheiten definiert werden, im Genaueren Zoll $\rightarrow$ cm.
2. Schreibe nun eine Funktion, welche eine Liste von Zahlen als Parameter nimmt, das größte Element in dieser bestimmt und es zusammen mit seinem Index in der Liste auf der Konsole ausgibt. Achte dabei auf Sonderfälle.

```python
def in_to_cm(inch):
    return inch * 2.54
```

```python
def print_max(lst):
    if not lst:
        print('Die Liste ist leer, es gibt kein Maximum')
        return

    index = -1
    maximum = 0
    for k, e in enumerate(lst):
        if e > maximum:
            index = k
            maximum = e

    print(f'Das Maximum der Liste ist {maximum}')
```

3.2 Bildschirmdiagonale

Nun soll die Größe eines Bildschirms im metrischen System anhand des Seitenverhältnisses und der Bildschirmdiagonale in Zoll berechnet werden. Schreibe dafür eine Funktion, welche die Bildschirmdiagonale in Zoll und das Seitenverhältnis (Breite:Höhe) entgegennimmt und die Breite und Höhe in Zentimetern zurückgibt.
Hinweis: Es kann die Konvertierungsfunktion aus der vorherigen Aufgabe wiederverwendet werden.
Die mathematischen Zusammenhänge sind in Abbildung 1 illustriert.

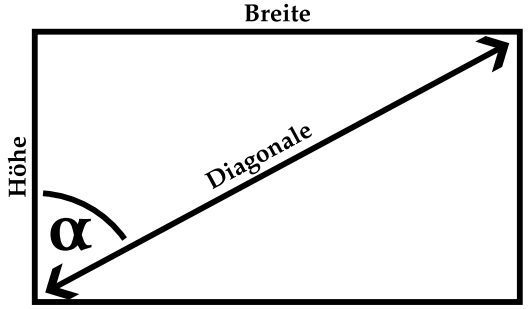

Abbildung 1: Mathematische Zusammenhänge der Bildschirmdiagonale

Hinweis: Sinus, Kosinus und Tangens sind mit `math.sin()`, `math.cos()`, `math.tan()` *verfügbar bzw.* `math.asin()`, `math.acos()` *und* `math.atan()` *für Arcus Sinus, etc.. Diese Funktionen können genutzt werden, nachdem sie importiert wurden. Beispiel:*

```python
import math
```

```

radians = math.pi * 0.5
value = math.sin(radians) # => 1.0
```

```python
import math

def abmessungen(inch, aspect_ratio):
    alpha = math.atan(aspect_ratio)
    cm = in_to_cm(inch)
    return (cm * math.sin(alpha), cm * math.cos(alpha))

w, h = abmessungen(15.6, 16.0/9.0)
print(f'Der Bildschirm hat eine Größe von {w:.2f} {h:.2f} cm')
```

3.3 Farbkonvertierung (Bonus)

Abschließend soll noch die Konvertierung von Farben aus dem HSV- in den RGB-Farbraum implementiert werden (mehr Infos auf Wikipedia). Die Werte einer Farbe im HSV-Format (**H**ue, **S**aturation, **V**alue) werden in den Wertebereichen $H \in [0, 360]$, $S \in [0, 1]$ und $V \in [0, 1]$ angegeben. Die Verwendung des HSV-Formats wird zum Beispiel für die Generierung von Farben mit gleichem Farbtonabstand benötigt. Schreibe eine Funktion, welche die Parameter h, s und v hat und ein Dreier-Tuple für die RGB-Farben zurückgibt.

$$h_i = [\frac{H}{60°}]$$
$$f = \frac{H}{60°} - h_i$$

$$p = V \cdot (1 - S)$$
$$q = V \cdot (1 - S \cdot f)$$
$$t = V \cdot (1 - S \cdot (1 - f))$$

$$(r, g, b) = \begin{cases} (q, V, p) & h_i = 1 \\ (p, V, t) & h_i = 2 \\ (p, q, V) & h_i = 3 \\ (t, p, V) & h_i = 4 \\ (V, p, q) & h_i = 5 \\ (V, t, p) & sonst \end{cases} \quad (1)$$

Die resultierenden RGB-Farben befinden sich im Wertebereich [0,1].
Hinweis: Mit der Funktion math.floor() können Fließkommazahlen abgerundet werden.

```python
def hsv2rgb(h, s, v):
    h_i = h // 60
    f = h / 60 - h_i
    p = v * (1 - s)
    q = v * (1 - s * f)
    t = v * (1 - s * (1 - f))
    match h_i:
        case 1:
            return (q, v, p)
        case 2:
            return (p, v, t)
```

```
        case 3:
            return (p, q, v)
        case 4:
            return (t, p, v)
        case 5:
            return (v, p, q)
        default:
            return (v, t, p)
```

Alternativ können statt `match` auch mehrere `if`-Fällen genutzt werden.