Übungszettel 4c - Fehlersuche

Aufgabe 1: Syntaxfehler

Betrachten Sie die folgenden Codeschnipsel. Probieren Sie sie aus und korrigieren Sie die Fehler.

(a) `timeA.py`

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_int)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

(b) `timeB.py`

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait"

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_str)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

(c) `timeC.py`

```
current_time_str = input("What is the "current time" (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_str)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

(d) `timeD.py`

```
str_time = input("What time is it now?")
str_wait_time = input("What is the number of hours to wait?")
time = int(str_time)
wai_time = int(str_wait_time)

time_when_alarm_go_off = time + wait_time
print(time_when_alarm_go_off)
```

(e) `timeE.py`

```
n = input("What time is it now (in hours)?")
n = imt(n)
m = input("How many hours do you want to wait?")
m = int(m)
q = m % 12
print("The time is now", q)
```

(f) `timeF.py`

```
a = input('wpisz godzine')
x = input('wpisz liczbe godzin')
int(a)
int(x)
h = x//24
s = x % 24
a = a + s
print('godzina teraz', a)
```

(g) `timeG.py`

```
present_time = input("Enter the present time in hours:")
set_alarm = input("Set the hours for alarm:")
int (present_time, set_time, alarm_time)
alarm_time = present_time + set_alarm
print(alarm_time)
```

Aufgabe 2: Hier stimmt doch was nicht

Das folgende Programm soll den *Body-Mass-Index (BMI)* für Teilnehmer einer Studie berechnen. Der BMI berechnet sich aus:

$$\frac{\text{Gewicht}[kg]}{\text{Körpergröße}^2[m]}$$

```
patients = [[70,1.8], [80, 1.9], [150, 1.7]]

def calculate_bmi(weight, height):
    return weight/(height **2)

for patient in patients:
    weight, heigth=patients[0]
    bmi = calculate_bmi(height, weight)
    print("Patient's BMI is:", bmi)
```

(a) Laden Sie den Quellcode `bmi.py` von der Vorkursseite (http://vorkurs.informatik.uni-frankfurt.de) herunter.

(b) Lassen Sie das Programm laufen. Falls der Interpreter Fehler anzeigt, korrigieren Sie diese, sodass Sie ein lauffähiges Programm erhalten

(c) Die Daten der Teilnehmer sind in der Liste `patients` gespeichert. Kann die Programmausgabe richtig sein?

Aufgabe 3: Mäuse

Folgendes Programm soll eine Mäusepopulation simulieren.

```
# Eingabe
print('Gib die Werte der Ausgangspopulation ein:')
jung = input('Anzahl junger Mauese: ')
erwachsen = input('Anzahl erwachsener Maeuse: ')
alt = ('Anzahl alter Maeuse: '
#Verarbeitung
hilf = erwachsen*4 + alt*2
alte = erwachsen // 3
erwachen = jung // 2
jung = hilf
#Ausgabe:
print('Die berechneten Populationswerte sind:')
print('Anzahl junger Maeuse: ', jung)
print('Anzahl erwachsenr Maeuse:', erwachsen)
print('Anzahl alter Maeuse:', alt)
```

(a) Laden Sie den Quelltext `debugMauese.py` von der Vorkursseite herunter.

(b) Korrigieren Sie das Programm, sodass es ohne Fehlermeldungen läuft.

(c) Testen Sie das Programm. Was soll berechnet werden? Stimmt die Ausgabe?

(d) Ändern Sie das Programm so, dass es auch bei Nutzereingaben die nicht dem vorgesehenen Format entsprechen (ganze Zahlen > 0) nicht abstürzt.

Aufgabe 4: Programmieraufgabe: Euklidischer Algorithmus

Der euklidische Algorithmus ist ein Verfahren den größten gemeinsamen Teiler (ggT) zweier Zahlen zu ermitteln.

Man teilt die größere durch die kleinere Zahl. Geht die Division auf, ist der Divisor der ggT. Geht die Division nicht auf, bleibt ein Rest. Dieser Rest ist der neue Divisor. Der alte Divisor wird zum Dividenden. Nach endlich vielen Schritten erhält man den ggT.

(a) Schreibe eine Funktion, die den ggT zweier übergebener Zahlen berechnet.

(b) Schreibe ein Programm, das den Benutzer natürliche Zahlen eingeben lässt und diese in einer Liste speichert.

- Überlege dir, wie der Benutzer dem Programm mitteilen soll, dass er/sie keine weiteren Werte mehr eingeben möchte.
- Ist die Eingabe vollständig, soll das Programm den ggT aller benachbarten Zahlen in der Liste berechnen und diese ausgeben.
- nutze dafür die in (a) programmierte Funktion, indem du sie als Modul importierst.

Beispiel Wenn die Zahlen 27, 9, 33, 1987 und 3195 eingegeben werden, soll das Programm folgende Ausgabe haben:
ggT von 27 und 9 : 9
ggT von 9 und 33 : 3
ggT von 33 und 1987 : 1
ggT von 1987 und 3195 : 1

Alternativ kannst du auch das Programm `debugGgTProg.py` und die Funktion `debugEuklid.py` von der Webseite herunterladen und debuggen.

Aufgabe 5: Programmieraufgaben

Falls Sie weitere Programmieraufgaben brauchen zum Üben:

(a) Schreibe ein Programm, das alle Zahlen zwischen 2000 und 3200 (einschließlich) die ein Vielfaches von 7, aber nicht von 5 sind. Die Zahlen sollen in einer Zeile, durch Kommata getrennt ausgegeben werden.

(b) Schreibe ein Programm, das Werte nach folgender Formel berechnet:

$$Q = \sqrt{\frac{2 \cdot C \cdot D}{H}}$$

Dabei sind H und C Konstanten, mit den Werten $C = 50$ und $H = 30$. D ist eine Variable. Der Nutzer soll die Möglichkeit haben gleich mehrere Werte mit Kommata getrennt über die Konsole einzugeben. Die Ausgabe soll ebenfalls als durch Kommata getrennte Werte erfolgen.
Beispiel:
Für die Eingabe `100,150,180`, soll das Programm folgende Ausgabe liefern: `18,22,24`
Hinweis: Die Methode `split()` für `strings` könnte hilfreich sein bei der Verarbeitung der Eingabe.

Viel Erfolg!