Aufgabenblatt While-Loop

Pflicht: Aufgaben 1-3

Kür: restliche Aufgaben

Aufgabe 1

Gegeben seien folgende Programme. Schreiben Sie dazu Programme mit dem gleichen Verhalten, welche ohne for-Loops auskommen und stattdessen while-Loops nutzen.

```
a) for i in range(5):
       for j in range(5):
           print(i, j)


b) for i in range(5):
       for j in range(i):
           print(i, j)
```

Aufgabe 2

Schreiben Sie ein Programm, bei dem die Anwenderin/ der Anwender Strings eingeben soll. Diese werden allenfalls bearbeitet und dann in den meisten Fällen in einer Liste gespeichert.

Strings, die ein 'x' enthalten, werden dabei einfach übersprungen und entsprechend auch nicht in die Liste aufgenommen. Bei Strings, die mindestens zwei 'e' enthalten, soll der String nach dem ersten e in zwei Teilstrings aufgeteilt werden. Alle anderen Strings werden unverändert in die Liste aufgenommen. Wenn allerdings der String 'letztes' gelesen wird, wird das Vorgehen vorzeitig abgebrochen.

Am Ende des Programms soll die entstandene Liste in der Shell ausgegeben werden.

Nutzen Sie die Möglichkeiten von *continue* und *break* und die verschiedenen Methoden von Strings!

Aufgabe 3

Schreiben Sie eine Funktion collatz(n), die folgendes macht:

Solange n grösser als 1 ist, wird n ersetzt durch

- n//2 falls n gerade ist
- 3*n+1 sonst

Zudem wird das aktuelle n jeweils in der Shell ausgegeben. Testen Sie für verschiedene Zahlen, was da passiert. Testen Sie auch die folgenden Zahlen: 2048, 871, 6171

Zusatzinformation: Die sogenannte Collatz-Vermutung besagt, dass dieser Prozess für alle natürlichen Zahlen n irgendwann endet. Lothar Collatz hat die Vermutung 1937 aufgestellt, ohne Sie jedoch beweisen zu können. Die Vermutung wurde auch in den darauffolgenden 83 Jahren nicht gelöst, weswegen diese Vermutung auch als Collatz Problem bekannt ist.

Aufgabe 4

Schreiben Sie ein Programm, das ähnlich wie jenes aus der Präsentation zufällige Additionsaufgaben stellt und solange neue Rechnungen erstellt und das Ergebnis abfragt, bis das erste Mal eine falsche Lösung durch den User eingegeben wird. Nutzen Sie dazu die Konstruktion `while True`

Aufgabe 5

Erweitern Sie den fiesen Rechnungstrainer so, dass das Kind statt der Lösung auch «111» eingeben kann und dann einfach die nächste Aufgabe gestellt wird, ohne dass das als Fehler gewertet wird.

Aufgabe 6

Schreiben Sie ein Programm netter_rechnungstrainer, der Multiplikationsaufgaben aus dem kleinen 1x1 stellt, ähnlich den anderen Rechnungstrainern. Am Ende wird die Anzahl falscher Lösungen ausgegeben und das Programm läuft so lange, bis 10 Aufgaben richtig gelöst wurden. Für jede Aufgabe hat das Kind drei Versuche.

Aufgabe 7

Schreiben Sie ein Programm mit einer Funktion rateZahl(n), bei dem in der Funktion eine Zufallszahl zwischen 1 und n erzeugt wird und in welcher der User so lange Zahlen eingeben muss, bis er die Zahl richtig erraten hat. Als Hilfestellung werden dem User bei falschen Versuchen jeweils Tipps gegeben, ob die gesuchte Zahl grösser oder kleiner als die eingegebene Zahl ist. Sobald die Zahl erraten wurde, wird dem User ein sinnvolles Feedback gegeben, das auch beinhaltet, wie viele Versuche er oder sie benötigt hat.

Anschlussfrage: Welches ist die beste Strategie für den User, um im Schnitt möglichst rasch die richtige Zahl zu erraten?

Aufgabe 8*

Kopieren Sie ihr Programm aus Aufgabe 3 und machen Sie folgendes. Entfernen Sie zuerst die print-Statements und schreiben Sie anschliessend eine zweite Funktion laengsteCollatzFolge(n), dass für alle Zahlen von 1 bis n jeweils diejenige Ausgangszahl sucht und zurückgibt, für die es am meisten Schleifendurchläufe braucht, bis man bei 1 landet. Passen Sie ihre ursprüngliche Funktion collatz(n) so an, dass Sie die neue Aufgabe erfüllen kann.