SS 2014

Prof. Dr. Margarita Esponda

ALP2: Objektorientierte Programmierung

1. Übungsblatt

Ziel: Auseinandersetzung mit logischen Ausdrücken, höheren Datenstrukturen, Schleifen und Funktionen in Python.

1. Aufgabe (4 Punkte)
Programmieren Sie eine Python-Funktion, die mit Hilfe einer **for**-Schleife nach Eingabe einer Liste mit Zahlen die Summe aller Zahlen der Liste als Ergebnis der Funktion zurückgibt.

2. Aufgabe (6 Punkte)
Schreiben Sie eine Python-Funktion **echtTeiler**, die bei Eingabe einer natürlichen Zahl **n** die Liste aller Teiler von **n** berechnet. Die Zahl **n** ist kein echter Teiler von **n** und soll deswegen nicht in die Ergebnisliste vorkommen.

Anwendungsbeispiel:

```
>>> echtTeiler(250)
>>> [1, 2, 5, 10, 25, 50, 125]
```

3. Aufgabe (4 Punkte)
Zwei natürliche Zahlen (m, n) werden als "Befreundetes Zahlenpaar" bezeichnet, wenn jede Zahl gleich der Summe der echten Teiler der anderen Zahl ist. Schreiben Sie eine Funktion in Python, die bei Eingabe zweier natürlicher Zahlen entscheidet, ob die Zahlen befreundet sind oder nicht.

4. Aufgabe (8 Punkte)
Die Collatz-Folge ist eine interessante Folge, die im Jahr 1937 von Lothar Collatz entdeckt worden ist.

Die **i**-te Collatz-Zahl wird mit der folgenden einfachen Definition berechnet:

$$C_i = \begin{cases} \dfrac{C_{i-1}}{2} & \text{wenn } C_{i-1} \text{ gerade} \\ C_{i-1} \cdot 3 + 1 & \text{wenn } C_{i-1} \text{ ungerade} \end{cases}$$

D.h., die Folge startet mit einer beliebigen Zahl **n**. Wenn **n** gerade ist, ist die Folgezahl gleich $\frac{n}{2}$ und wenn **n** ungerade ist, ist die Folgezahl gleich $(n \cdot 3 + 1)$.

Die Vermutung ist, dass egal mit welcher natürlichen Zahl gestartet wird, die Folge immer mit dem Zahlenzyklus 4, 2, 1 endet. Eine Vermutung, die bis jetzt noch nicht bewiesen worden ist.

a) Schreiben Sie eine Python-Funktion, die bei Eingabe einer beliebigen natürlichen Zahl **n** die Collatz-Folge berechnet und in eine Liste als Ergebnis der Funktion zurückgibt. Die Folge endet, wenn zum ersten mal die Zahl **1** erreicht wird.

Anwendungsbeispiel:

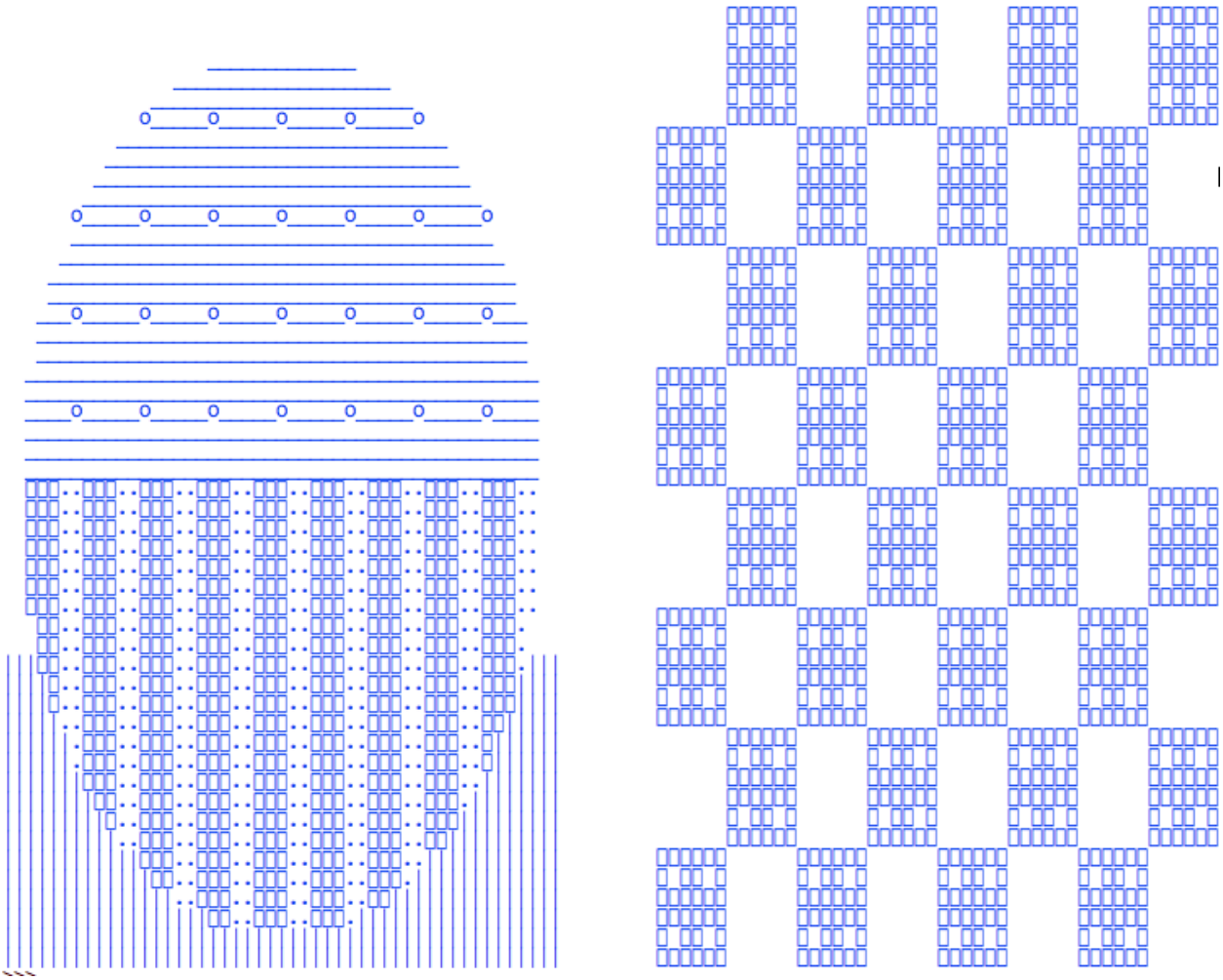
```
>>> collatz(10)
>>> [10, 5, 16, 8, 4, 2, 1]
```

b) Programmieren Sie eine **collatz_seqs** Funktion, die bei Eingabe einer natürlichen Zahl **n** die Collatz-Folge aller natürlichen Zahlen zwischen **1** und **n** berechnet und das Ergebnis als Liste von Listen zurückgibt. Schreiben Sie eine Hilfsfunktion **print_collatz** für eine übersichtlicher Ausgabe der Folgen.

Anwendungsbeispiel:

```
>>> print_collatz(collatz_seqs(12))
>>>
1 :  [1]
2 :  [2, 1]
3 :  [3, 10, 5, 16, 8, 4, 2, 1]
4 :  [4, 2, 1]
5 :  [5, 16, 8, 4, 2, 1]
6 :  [6, 3, 10, 5, 16, 8, 4, 2, 1]
7 :  [7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
8 :  [8, 4, 2, 1]
9 :  [9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
10 :  [10, 5, 16, 8, 4, 2, 1]
11 :  [11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
12 :  [12, 6, 3, 10, 5, 16, 8, 4, 2, 1]
>>>
```

5. Aufgabe (6 Punkte)
In dieser Aufgabe sollen die Funktionen **easter_egg** und **chessboard** definiert werden, die die unten stehenden Zeichenbilder produzieren.



Eine **print_char_picture** Funktion, die als Argument eine Funktion bekommt, wird vorgegeben. Diese Funktion darf nicht verändert werden. Innerhalb der **print_char_picture** Funktion bekommen die eingegebenen Funktionen eine **x, y** Koordinate und einen ganzzahligen **size** Wert, der der Seitenlänge eines Bildes entspricht, und entscheidet, welches Zeichen an einer bestimmten Position zurückgegeben wird.

Innerhalb der zu implementierenden Funktionen dürfen keine Schleifen verwendet werden. Die Lösung benötigt nur **if-else**-Anweisungen, logische Ausdrücke und arithmetische Operationen.

Die **print_char_picture** Funktion sowie zwei Beispielsfunktionen sind auf der Veranstaltungsseite unter Ressourcen herunter zu laden.

6. Aufgabe (6 Punkte)
Programmieren Sie eine Python-Funktion, die bei Eingabe einer natürlichen Zahl **n** alle befreundeten Zahlenpaare (a, b) berechnet, die kleiner **n** sind. Das bedeutet a und b sind befreundet und a<b<n.

Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Funktionalität der Funktionen darstellen.

2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

3) Kommentieren Sie Ihre Programme.

4) Verwenden Sie geeignete Hilfsvariablen und Hilfsfunktionen in Ihren Programmen.

5) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.

6) Schreiben Sie getrennte Test-Funktionen für alle 6 Aufgaben für Ihren Tutor.

7) Die Lösungen sollen in Papierform und elektronisch (KVV-Upload) abgegeben werden.