Schleifenübung

Prof. Dr. Robert Gold

Fakultät Informatik
Technische Hochschule Ingolstadt

Wintersemester 2019/20

Schleifen gehören zu den wichtigsten Kontrollstrukturen in der Programmierung. In diesem Dokument finden Sie einige Übungsaufgaben zu Schleifen. Zu jeder Aufgabe soll ein Python-Skript mit dem Aufbau

i. Einlesen der nötigen Eingabewerte von der Konsole (input) und Zuweisung an Variablen
ii. Berechnung der geforderten Ergebnisse mit Schleifen
iii. Ausgabe der geforderten Ergebnisse auf die Konsole (print)

erstellt werden.

While-Schleifen

Aufgabe 1 (Aufgabe 3 in Kapitel 1.6)

Eingabe:	Ganze Zahl n	mit n > 0
Ausgabe:	Ganze Zahl :	Fibonaccizahl fib(n) Die Fibonaccizahl ist definiert durch: fib(0) = 0, fib(1) = 1, fib(k) = fib(k-1) + fib(k-2) *unter Verwendung einer while-Schleife*

Aufgabe 2 (Aufgabe 4 in Kapitel 1.6)

Eingabe:	Ganze Zahl n	mit n > 0
Ausgabe:	Ganze Zahl :	n-tes Glied der Zahlenfolge 1, 2, 4, 7, 11, 16, 22, … *unter Verwendung einer while-Schleife*

Aufgabe 3 (Aufgabe 6 in Kapitel 1.7)

„In der nächsten Aufgabe lernen wir einen besonderen Frosch kennen, so wie ihn sich nur Mathematiker ausdenken können. Besonders seine Art, eine Straße zu überqueren, macht es zweifelhaft, ob er in der realen Welt lange überleben könnte. Er überquert eine 2,50 Meter breite Straße wie folgt: Mit dem ersten Sprung legt er die erstaunliche Distanz von einem Meter zurück, dann springt er wegen zunehmender Erschöpfung mit jedem weiteren Schritt immer nur noch halb so weit wie vorher. Die Entfernung, die er dabei zurücklegt, berechnet sich also als Summe der Werte 1+0,5+0.25+0,125 und so weiter. […] Versuchen Sie […] herauszubekommen, ob der Frosch es auf die andere Straßenseite schafft."
(Klein 2018, Seite 84-85)

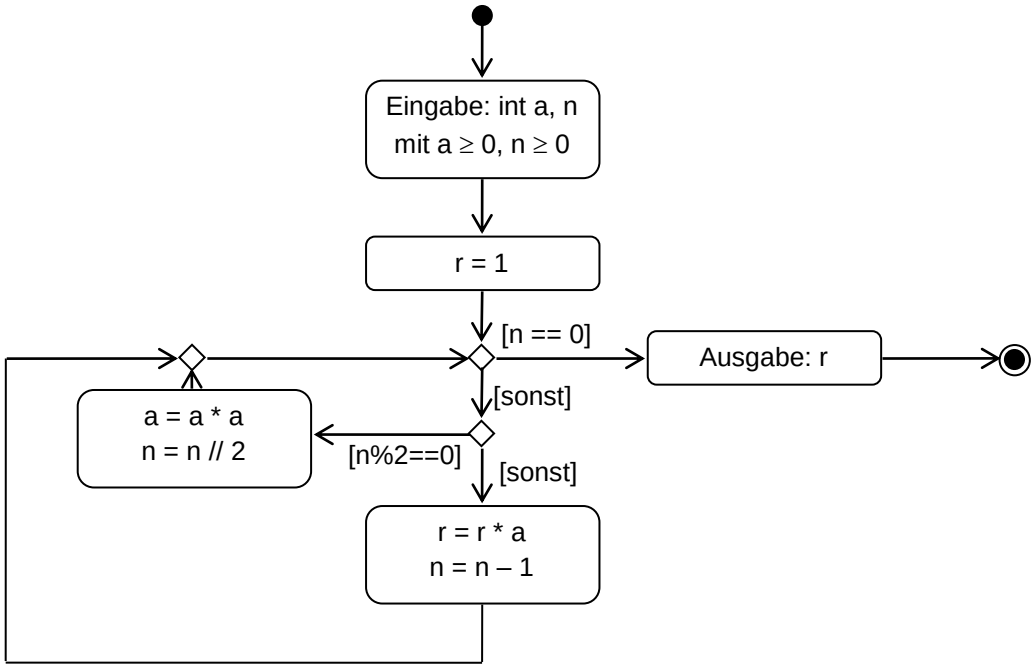
Eingabe:	Ganze Zahl n	mit n > 0
Ausgabe:	Fließkommazahl :	Distanz, die der Frosch mit n Sprüngen überwindet *unter Verwendung einer while-Schleife*

Aufgabe 4 (Aufgabe 2 in Kapitel 1.8)

Eingabe:	Ganze Zahl n	mit n > 0
Ausgabe:	Ganze Zahl :	Summe der Zahlen von 1 bis n *unter Verwendung einer fußgesteuerten while-Schleife*

Aufgabe 5

Gegeben sei das folgende Ablaufdiagramm. Überlegen Sie sich zuerst mit einer Wertetabelle für a = 2 und n = 5, was das Algorithmus macht. Schreiben Sie dann ein gleichwertiges Python-Skript.



Aufgabe 6

Übertragen Sie das Ablaufdiagramm zur Nullstellenbestimmung in den Folien in ein gleichwertiges Python-Skript.

Eingabe:	keine	
Ausgabe:	Fließkommazahl :	Nullstelle des Polynoms im Intervall zwischen 0 und 1.5 *unter Verwendung einer while-Schleife*

For-Schleifen

Aufgabe 7 (Aufgabe 3a in Kapitel 1.8)

Eingabe:	Ganze Zahl n	mit n > 0
Ausgabe:	Ganze Zahl :	Summe der Zahlen von 1 bis n *unter Verwendung einer for-Schleife*

Aufgabe 8 (Aufgabe 3b in Kapitel 1.8)

Eingabe:	Ganze Zahl n	mit n > 0
Ausgabe:	Ganze Zahl :	Fakultät n! = 1 · 2 · 3 · … · n *unter Verwendung einer for-Schleife*

Aufgabe 9 (Aufgabe 3c in Kapitel 1.8)

Eingabe:	Ganze Zahl n	mit n > 0
Ausgabe:	Ganze Zahl :	n-tes Glied der Zahlenfolge 1, 2, 4, 7, 11, 16, 22, … *unter Verwendung einer for-Schleife*

Schleifen mit zwei Schleifenbedingungen

Schleifen mit zwei Schleifenbedingungen a und b können als while-Schleife

```
while a and b:
    …
```

erstellt werden. Falls eine Abbruchbedingung der Ablauf eines Zählers ist, z.B. i in range(0, n), kann auch eine for-Schleife

```
for i in range(0, n):
    if not b:
        break
    …
```

verwendet werden. Wenn man nach der Schleife herausfinden muss, welche Schleifenbedingung nicht mehr erfüllt war, braucht man eine Abfrage nach der Schleife, z.B.:

```
if not a:
    …
else:
    …
```

Aufgabe 10

Schreiben Sie ein Python-Skript, das prüft, ob die eingegebene Zahl n eine Primzahl ist. Testen Sie alle Zahlen i auf „n teilbar durch i“, solange bis eine Teiler gefunden. Es gibt zwei Schleifenbedingungen: Noch eine Zahl i zu testen und bisher kein Teiler gefunden.

Eingabe:	Ganze Zahl n	mit $n \geq 2$
Ausgabe:	Wahrheitswert :	True, falls n eine Primzahl ist False, falls n keine Primzahl ist *unter Verwendung einer while-Schleife*

Geschachtelte Schleifen

In vielen Aufgabenstellungen müssen zwei (oder mehr) Schleifen ineinander geschachtelt werden. Häufig kommt das bei tabellenartigen Datenstrukturen vor. In der äußeren Schleife durchläuft man die Zeilen und in der inneren Schleife die Spalten der Tabelle. Beispielsweise, wenn die Tabelle oder Matrix n Zeilen und m Spalten hat:

```
for i in range(0, n):
    for j in range(0, m):
        …
```

Aufgabe 11

Schreiben Sie ein Skript, das am Beispiel von n == 64 folgende Konsolausgabe erstellt:

```
****************************************************************
********************************
****************
********
****
**
*
```

In der ersten Zeile stehen n Sterne, in der zweiten Zeile n / 2 Sterne usw.

Eingabe:	Ganze Zahl n	mit n ist eine Zweierpotenz
Ausgabe:	Siehe oben	*unter Verwendung von zwei geschachtelten Schleifen*

Natürlich gibt es in Python auch ein Konstrukt, das ohne Schleife einen String bestehend aus einer vorgegebenen Anzahl von Sternen konstruiert. Ein solches Konstrukt soll aber nicht verwendet werden.

Aufgabe 12

Ändern Sie das Skript aus der letzten Aufgabe, sodass am Beispiel von n == 64 folgende Konsolausgabe entsteht:

```
****************************************************************
****************                                ****************
********                                            ********
****                                                    ****
**                                                        **
*                                                          *
*                                                          *
**                                                        **
****                                                    ****
********                                            ********
****************                                ****************
****************************************************************
```

Eingabe:	Ganze Zahl n	mit n ist eine Zweierpotenz
Ausgabe:	Siehe oben	*unter Verwendung von geschachtelten Schleifen*