Informatik Aufgaben

1. Erstelle ein Programm zur Berechnung der Summe der Zahlen von 1 bis n, z. B. $n = 100$.

2. Erstelle ein Programm, das die ersten 20 (z. B.) ungeraden Zahlen 1, 3, 5, ... ausgibt und deren Summe berechnet.

3. Auf dem Bildschirm soll ein Muster folgender Art

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

erzeugt werden.

4. Auf dem Bildschirm soll ein Muster folgender Art

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

erzeugt werden.

5. *Leibniz:* $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ...$

   Erstelle ein Programm zur Berechnung von π (auf 4 Stellen).

6. $$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + - \ \ldots \qquad (x \text{ im Bogenmaß, z. B. } 5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5)$$
   $$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + - \ \ldots$$
   $$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ \ldots$$

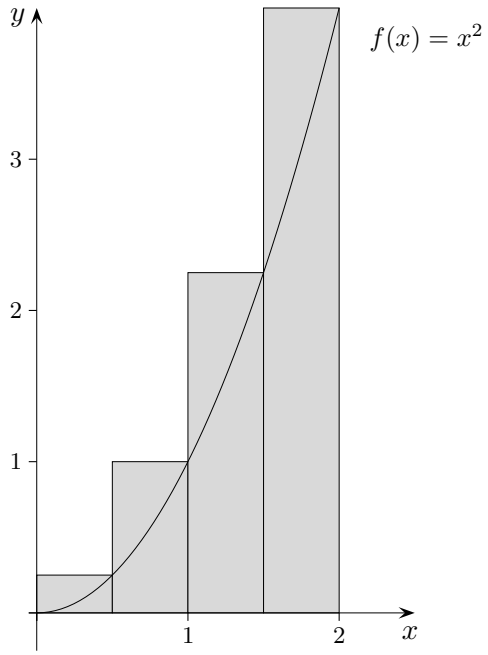
   Erstelle ein Programm zur Berechnung der Funktionswerte der Funktionen (auf 4 Stellen).

7. Fülle ein Array mit Zufallszahlen und suche das größte und kleinste Element.

8. Fülle ein Array mit Zufallszahlen und sortiere die Elemente der Größe nach.

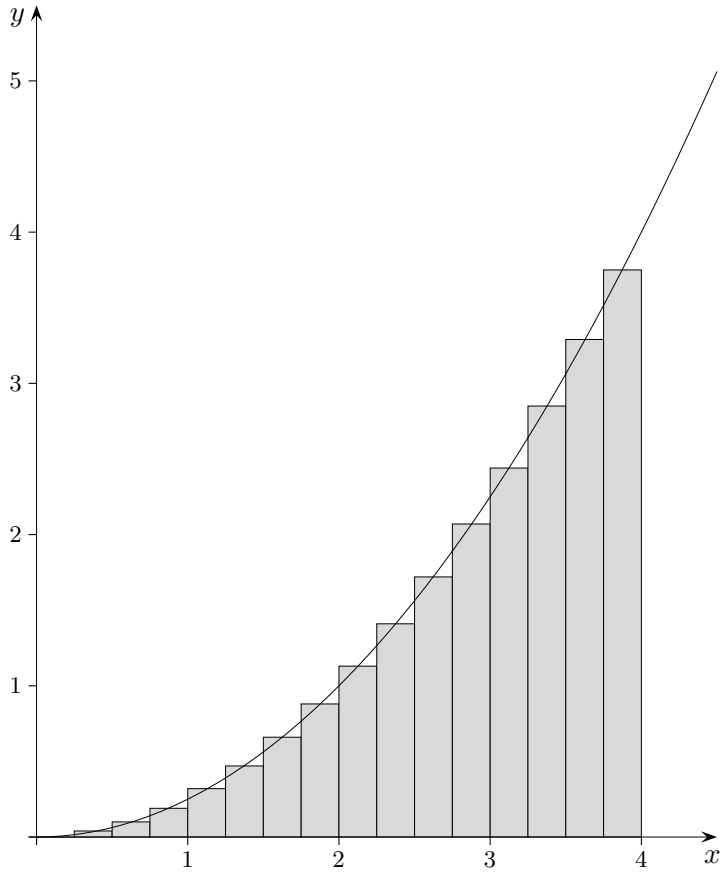
Flächenberechnung

9.

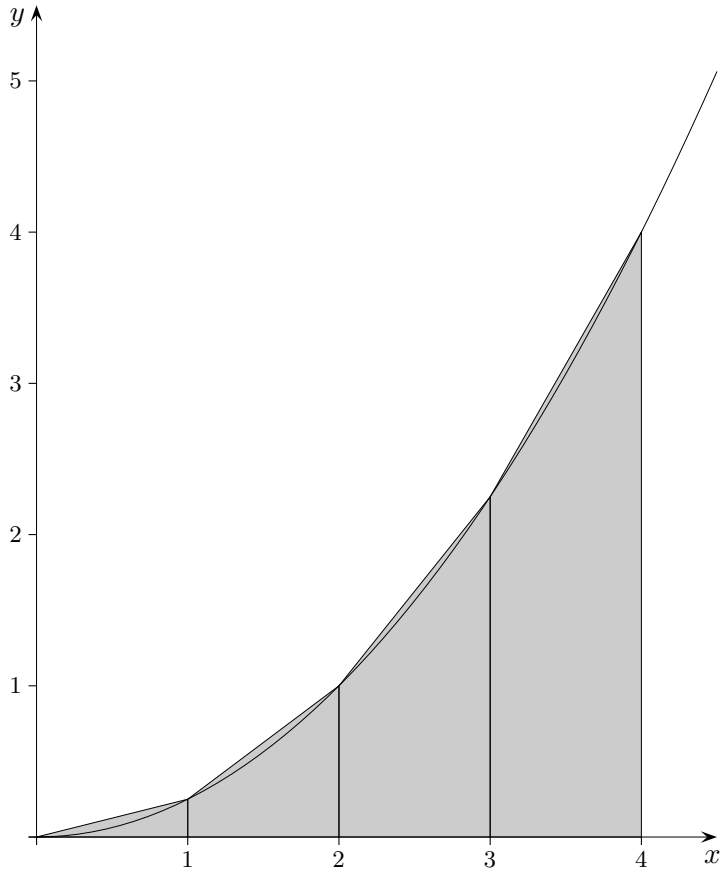


Die Fläche A unterhalb des Grafen von $f(x) = x^2$ in den Grenzen von 0 bis 2 wird hier durch eine Obersumme mit $n = 4$ Unterteilungen angenähert.
Entwickle ein Programm zur Berechnung der Obersumme für $n = 4, 8, 16, \ldots$
Berechne A auf drei Stellen genau.

Numerische Integration

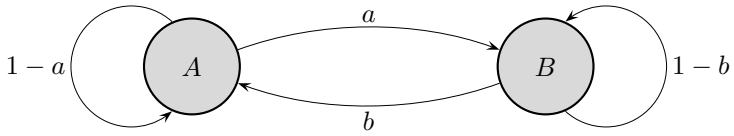


Numerische Integration



Taxifahrt

10.



Ein Taxifahrer beginnt seine Tour am Morgen im Ort A.
Er wechselt die Orte mit den Wahrscheinlichkeiten $a = \frac{1}{3}$ und $b = \frac{1}{4}$.
Wie groß ist die Wahrscheinlichkeit P, dass das Taxi nach n Fahrten in A ist?
Ändert sich diese Wahrscheinlichkeit beim Start in B?

Entwickle ein Simulationsprogramm, mit dem die Fragen beantwortet werden können.
Ist P auch die Aufenthaltswahrscheinlichkeit?

Summe von Zufallsvariablen

11. Die Zufallsvariablen X_i, $i = 1, \ldots, 20$, nehmen jeweils mit gleicher Wahrscheinlichkeit die Zahlen von 1 bis 5 an. S sei die Summe der X_i.

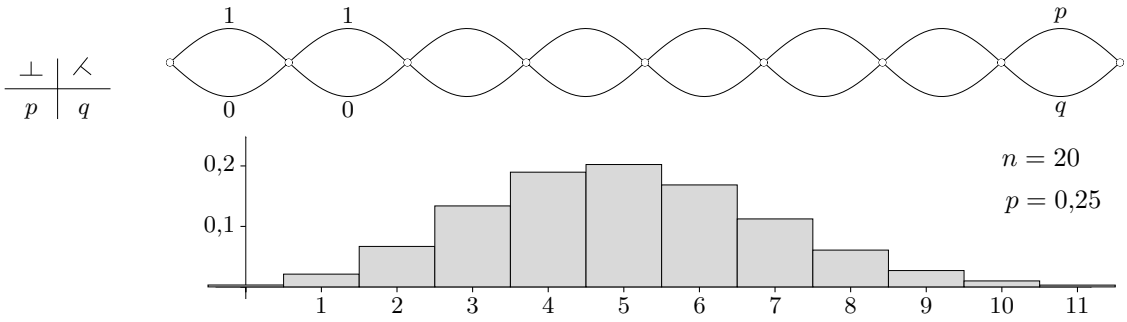
    Entwickle ein Simulationsprogramm, mit dem die Verteilung von S,
    also eine Tabelle der möglichen Ergebnisse von S mit den zugehörigen Wahrscheinlichkeiten (hier relativen Häufigkeiten) grafisch dargestellt wird.

    Variiere die Simulation.

Bernoulli-Kette Simulation

Zufallsexperimente, die nur genau zwei Ergebnisse haben können, heißen Bernoulli-Versuche.
Wird ein solches Experiment unter gleichen Bedingungen n-mal wiederholt, so liegt eine Bernoulli-Kette der Länge n vor.
Beispiele: n-maliges Werfen einer Reißzwecke, Folge von Geburten (Junge/Mädchen),
Testen von Glühbirnen mit den Einzelergebnissen defekt/nicht defekt.

Zur Unterscheidung wird ein Ausgang des Einzelversuchs mit 1 (Treffer), der andere mit 0 bezeichnet, die zugehörigen Wahrscheinlichkeiten mit p (Trefferwahrscheinlichkeit) und q.




```
from random import *

n=20            # Länge der Bernoulli-Kette, zunächst p=0.5
N=1000          # Anzahl der Wiederholungen eines Bernoulli-Versuchs

L = [0 for k in range(n+1)]

for i in range(N):
   anz=0                        # Anzahl der Treffer (Einsen)
   for j in range(n):
      anz = anz+randint(0,1)

   L[anz]=L[anz]+1/N

print(L)
```

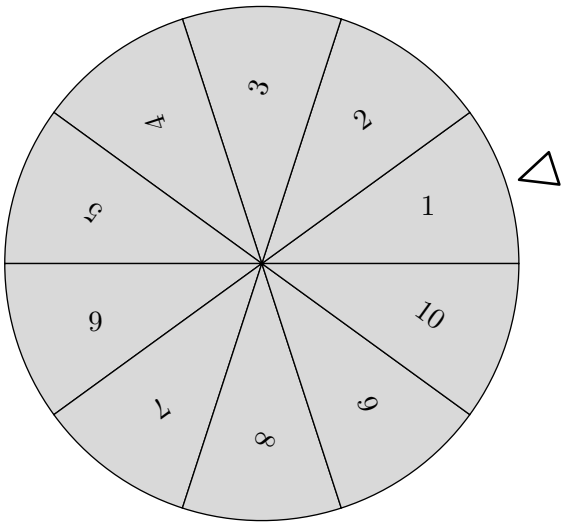
12. Ändere das Programm so ab, dass beliebige Wahrscheinlichkeiten p zugelassen sind. Erzeuge eine grafische Ausgabe.

```
from tkinter import*
master=Tk()
w=Canvas(master,height=600,width=1000,bg='white')
w.pack()

w.create_rectangle(10,300,20,300-80,fill='blue',outline='blue')
```

13. Fülle ein Array mit monoton wachsenden Zufallszahlen und entwickle ein Suchprogramm, das mit möglichst wenigen Vergleichen überprüft, ob eine vorgegebene Zahl im Array vorhanden ist.

14.



Wie oft ist ein Glücksrad mit den Zahlen von 1 bis n zu drehen, bis ein *vollständiger Satz* vorliegt, d. h. dass alle Zahlen mindestens einmal erschienen sind? Finde die Antwort durch Simulation.

15. Random Walk
Bei dieser Irrfahrt auf einer Geraden, bzw. einer Ebene (quadratisches Gitter) wird jeweils einer der zwei bzw. vier Nachbarn zufällig ausgewählt. Die Schrittlänge bleibt gleich.
Simuliere den Prozess.
Erzeuge eine Grafik, in der die Endpunkte nach einer konstanten Schrittzahl (z. B. $n = 100$) dargestellt sind.

16. Entwerfe eine rekursive Prozedur, die eine Schar ineinandergeschachtelter (konzentrischer) Kreise erzeugt.

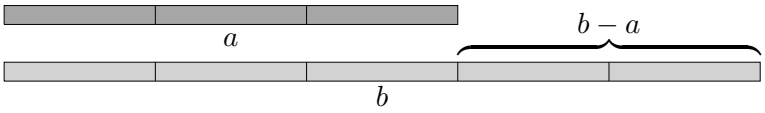
17. Gib begründet an, was jeweils mit folge(6) ausgegeben wird.

    a)
    ```
    def folge(n):
        if n>0:
            print(n)
            folge(n-1)
    ```

    b)
    ```
    def folge(n):
        if n>0:
            folge(n-1)
            print(n)
    ```

18. Schreibe eine rekursive Funktion ggT(n, m), die zu zwei gegebenen Zahlen den größten gemeinsamen Teiler ausgibt, z. B. $\text{ggT}(8, 12) = 4$.
    Die Idee dieser Rekursion ist im folgenden Problem enthalten.

    Stell Dir vor, du hast zwei Stäbe der Länge a und b, wobei a und b natürliche Zahlen sind. Du möchtest die Stäbe in gleichgroße Stücke zersägen und zwar so, dass die Länge d der Stücke möglichst groß.



19. Entwickle ein Programm, das einen beliebigen Bruch in der nicht kürzbaren Version ausgibt.

20. Schreibe ein rekursives Programm für die

    a) Fibonacci-Folge $1, 1, 2, 3, 5, 8, 13, \ldots$,

    b) Fakultät von n, z. B. $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$,

    c) Binomialkoeffizienten (Anzahl der k-elementigen Teilmengen)
    $$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \quad \binom{n}{n} = 1, \quad \binom{n}{1} = n,$$

    d) Ausgabe aller Permutationen von n Elementen,

    e) Erzeugung aller 0/1-Folgen der Länge n.

```
 OpenWindow(0, 0, 0, 800, 600, "Fenster Zeichnung",
                    #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 StartDrawing(WindowOutput(0))
   x1=400
   y1=300
   x2=400
   y2=300
   d=10

     Circle(400, 300, 200, RGB(202,255,112))
             For n = 1 To 100

                zufall.l=Random(3)
                            DrawText(100,100,Str(zufall))
                   If zufall=0
                        x2= x1+d
                   EndIf

                   If zufall=1
                        x2= x1-d
                   EndIf

                   If zufall=2
                        y2= y1+d
                   EndIf

                   If zufall=3
                        y2= y1-d
                   EndIf


                LineXY(x1,y1,x2,y2,RGB(Random(250), Random(250), Random(250)))
                     Delay(100)
                x1=x2
                y1=y2
            Next n



    Repeat
      Ereignis.i = WaitWindowEvent()
    Until Ereignis = #PB_Event_CloseWindow
    StopDrawing()
End



;  Link Farbtabelle    http://www.farb-tabelle.de/de/farbtabelle.htm
;  Sqr(Zahl)  Quadratwurzel
```

Federschwingung

Die Grundgleichung

$$F = m \cdot a$$

liefert bei konstanter Masse für beliebige Kräfte die Beschleunigung:

$$a = \frac{F}{m}$$

Die Kraft einer Feder ist in einem gewissen Bereich der Auslenkung proportional und entgegengerichtet:

$$F = -D \cdot s$$

Simuliere die Schwingung einer Federwaage. Erzeuge ein Weg-Zeit-Diagramm.
Die die Bewegung beeinflussende Gesamtkraft setzt sich aus der Erdanziehungskraft und der Federkraft zusammen:

$$F_{\text{gesamt}} = m \cdot g - D \cdot s$$

Konstanten und Anfangsbedingungen festlegen,

Wiederhole:

1. Kraft F berechnen, $F_{\text{gesamt}} = m \cdot g - D \cdot s$
2. Geschwindigkeit berechnen, $v_{\text{neu}} = v_{\text{alt}} + \frac{F}{m} \cdot \Delta t$
3. Weg berechnen, $s_{\text{neu}} = s_{\text{alt}} + v \cdot \Delta t$

Variation: Schwingung mit Dämpfung

Planetenbahn

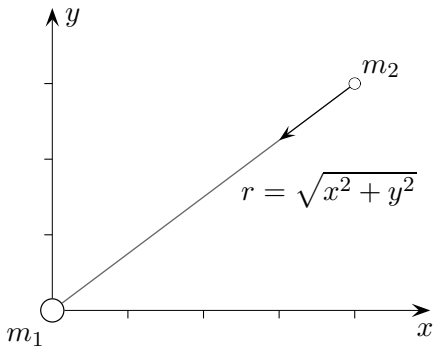
Nach dem allgemeinen Gravitationsgesetz von Newton ist die Kraft, die auf einen um die Sonne kreisenden Planeten wirkt:

$$F = f\,\frac{m_1\,m_2}{r^2}$$

Für eine Bewegung in der Ebene kann die Kraft für jeden Punkt $P(x \mid y)$ vektoriell ermittelt werden. Die Sonne befindet sich im Koordinatenursprung.

$$\begin{pmatrix} F_x \\ F_y \end{pmatrix} = f\,\frac{m_1\,m_2}{r^2}\begin{pmatrix} x \\ y \end{pmatrix}^0, \qquad f \text{ negativ},\ \vec{x} = r\cdot\vec{x}^{\,0}$$

$$= f\,\frac{m_1\,m_2}{r^3}\begin{pmatrix} x \\ y \end{pmatrix}$$



$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} = \frac{1}{m_2}\begin{pmatrix} F_x \\ F_y \end{pmatrix}$$

$$= f\,\frac{m_1}{r^3}\begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} + \Delta t\begin{pmatrix} a_x \\ a_y \end{pmatrix}$$

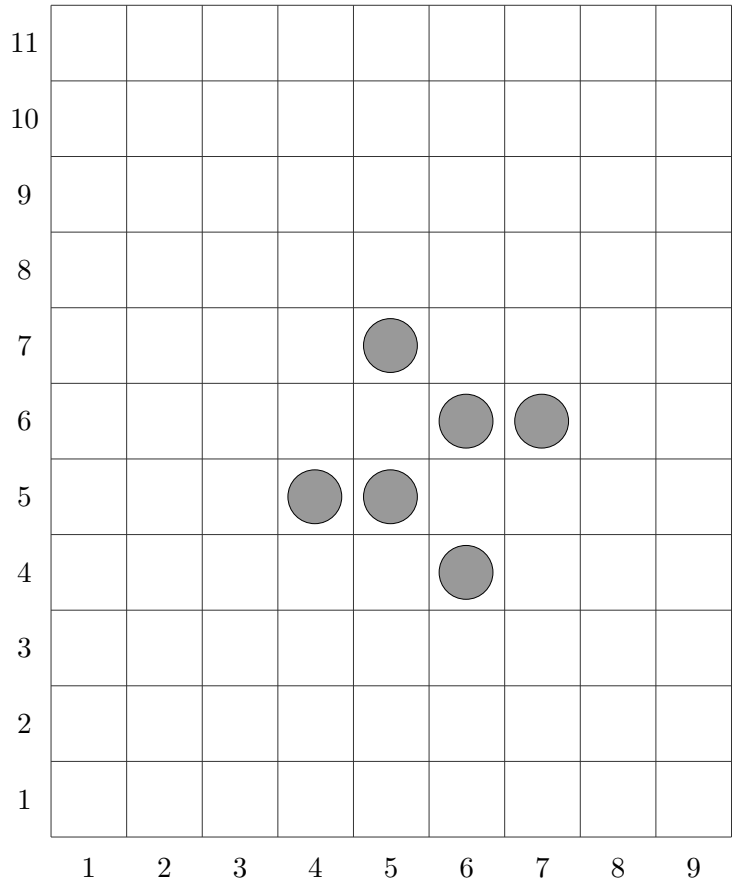
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \Delta t\begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

fm_1 kann zusammengefasst werden zu (z. B.) $k = -9000$. Die Planetenmasse m_2 ist unerheblich.
$\Delta t = 0.08$
$v_x = 0$, $v_y = 10$
Für die Grafik ist der Ursprung zu verschieben.

wiederhole

$r = \sqrt{x^2+y^2}$		Entfernung berechnen
$a_x = k\cdot x/r^3$		Beschleunigung
$a_y = k\cdot y/r^3$		
$v_x = v_x + a_x\cdot\Delta t$		Geschwindigkeit
$v_y = v_y + a_y\cdot\Delta t$		
$x = x + v_x\cdot\Delta t$		Position
$y = y + v_y\cdot\Delta t$		

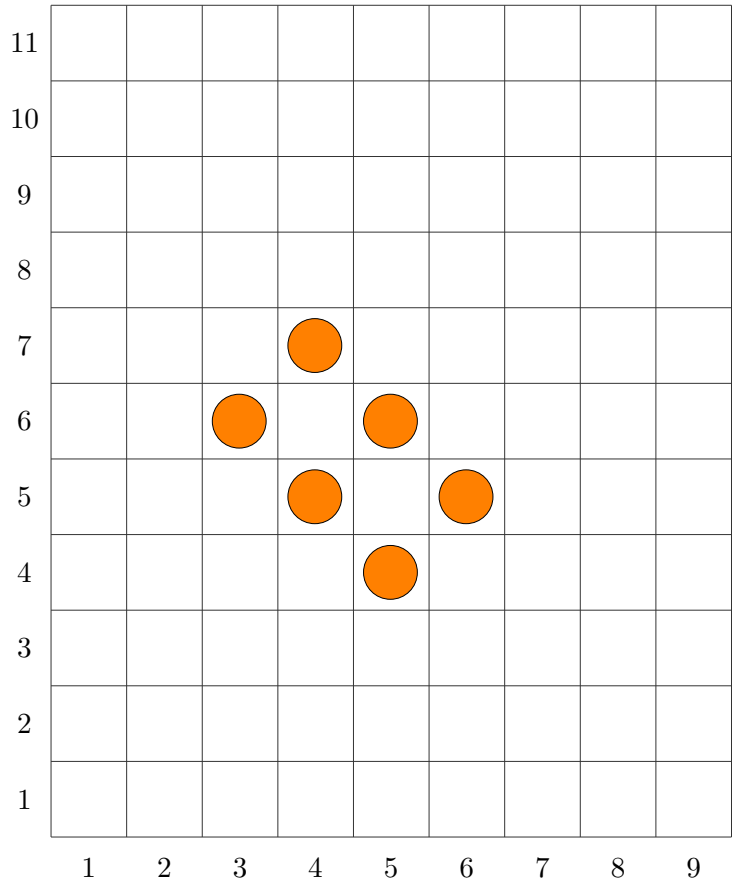
Game of Life



Lebewesen verteilen sich auf einem unbegrenzten, karierten Feld.
Je nach Nachbarschaft bleiben sie am Leben, sterben oder bringen neues Leben hervor.
Game of Life kann als zellulärer Automat angesehen werden, bei dem der Zustand einer Zelle vom eigenen Zustand und von dem der Nachbarzellen abhängt. Das Spiel wurde 1970 von John Conway entwickelt.

1. Das Lebewesen in einer Zelle überlebt genau dann, wenn es 2 oder 3 Nachbarn hat.
   (Bei keinem oder einem Nachbarn stirbt es aus Einsamkeit, bei 4 bis 8 wegen Überbevölkerung.)
2. Auf jeder freien Zelle, die genau an 3 bewohnte Zellen grenzt, entsteht neues Leben.

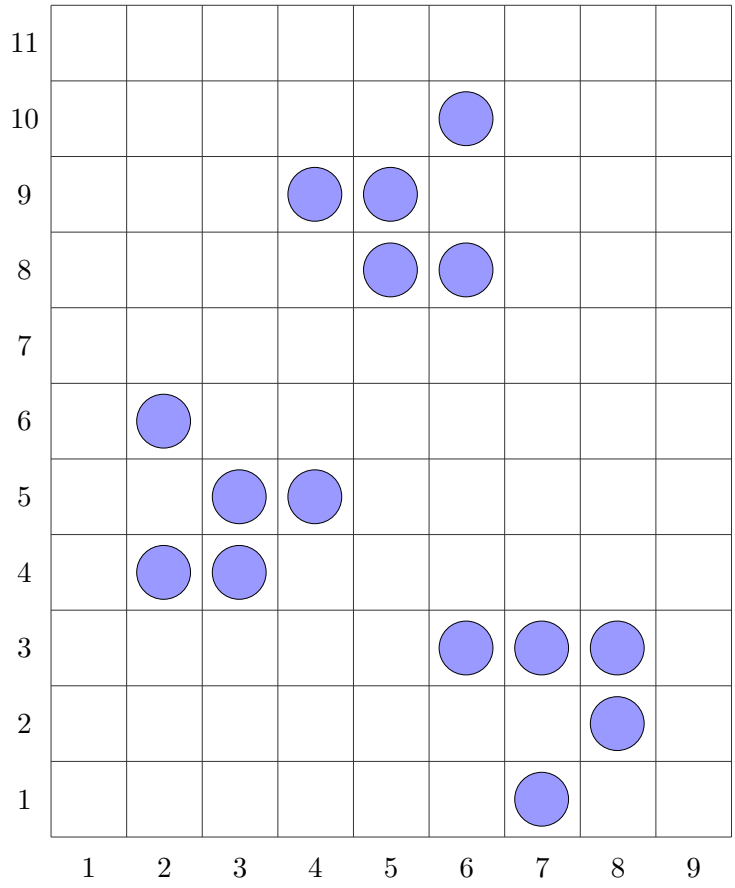
Game of Life



Lebewesen verteilen sich auf einem unbegrenzten, karierten Feld.
Je nach Nachbarschaft bleiben sie am Leben, sterben oder bringen neues Leben hervor.
Game of Life kann als zellulärer Automat angesehen werden, bei dem der Zustand einer Zelle vom eigenen Zustand und von dem der Nachbarzellen abhängt. Das Spiel wurde 1970 von John Conway entwickelt.

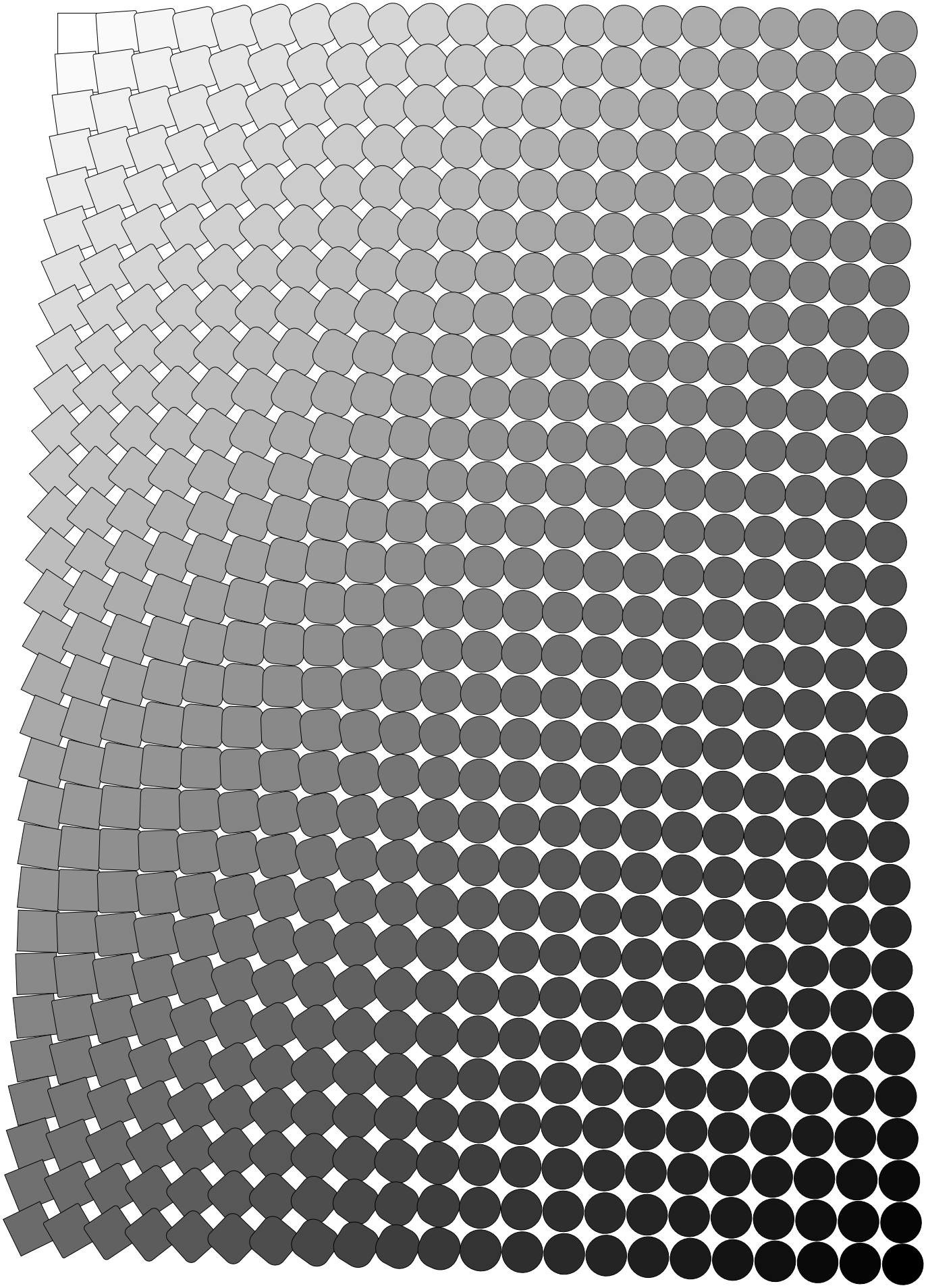
1. Das Lebewesen in einer Zelle überlebt genau dann, wenn es 2 oder 3 Nachbarn hat.
   (Bei keinem oder einem Nachbarn stirbt es aus Einsamkeit, bei 4 bis 8 wegen Überbevölkerung.)
2. Auf jeder freien Zelle, die genau an 3 bewohnte Zellen grenzt, entsteht neues Leben.

Game of Life



Lebewesen verteilen sich auf einem unbegrenzten, karierten Feld.
Je nach Nachbarschaft bleiben sie am Leben, sterben oder bringen neues Leben hervor.
Game of Life kann als zellulärer Automat angesehen werden, bei dem der Zustand einer Zelle vom eigenen Zustand und von dem der Nachbarzellen abhängt. Das Spiel wurde 1970 von John Conway entwickelt.

1. Das Lebewesen in einer Zelle überlebt genau dann, wenn es 2 oder 3 Nachbarn hat.
   (Bei keinem oder einem Nachbarn stirbt es aus Einsamkeit, bei 4 bis 8 wegen Überbevölkerung.)
2. Auf jeder freien Zelle, die genau an 3 bewohnte Zellen grenzt, entsteht neues Leben.



Morphing

Schreibe ein Programm zum Thema Morphing, z. B. kann der Übergang von Dreiecken zu Quadraten bearbeitet werden. Die vorige Seite dient lediglich der Erläuterung des Begriffs.

Warteschlange

Beim Herrenfriseur treffen im Schnitt 3 Kunden pro Stunde ein. Der Friseur benötigt für die Bedienung eines jeden Kunden genau 15 Minuten. Ermittle die mittlere Wartezeit und erstelle ein Histogramm der Wartezeiten mit Hilfe einer Simulation.

Entropie

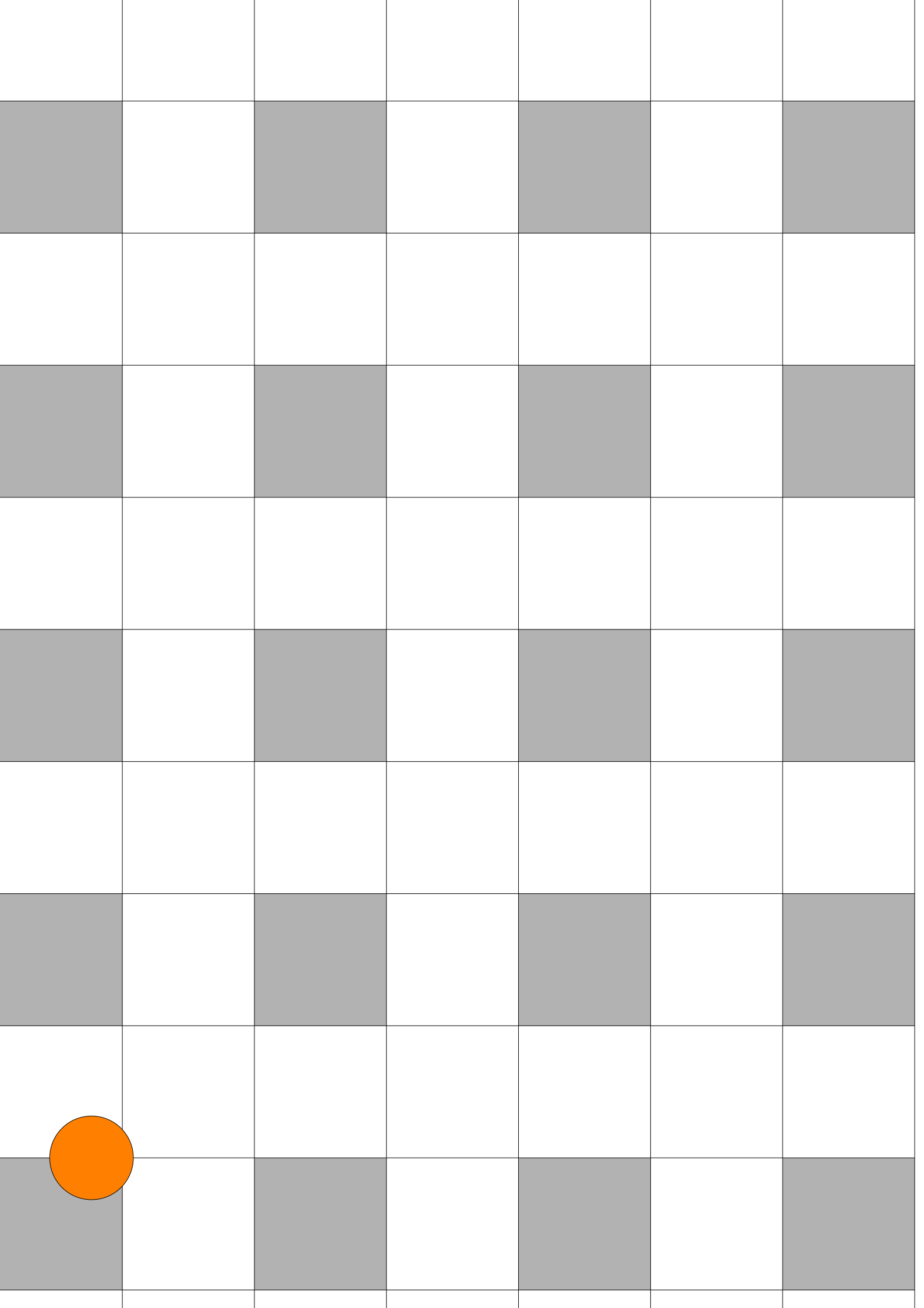
In einem von zwei Behältern befinden sich $n = 100$ nummerierte Kugeln. Wir erzeugen eine Zufallszahl aus dem Bereich $[1, 100]$. Die Kugel mit dieser Nummer wechselt den Behälter. Dieser Vorgang wird wiederholt. Der sich ändernde Betrag der Differenz der Kugelanzahlen im ersten und zweiten Behälter soll grafisch dargestellt werden.

Ich biete dir folgendes Spiel an:

Du wirfst eine 2-Cent-Münze, so dass sie auf einem Raster aus Quadraten landet, deren Seitenlänge $3\,cm$ beträgt.

Wenn deine Münze ganz in ein Quadrat fällt, gewinnst du 50 Cent, andernfalls bekomme ich von dir 10 Cent.

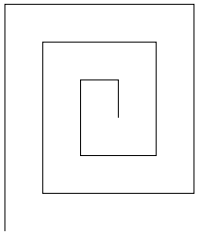
Finde deinen erwarteten Gewinn mit Hilfe einer Simulation heraus.



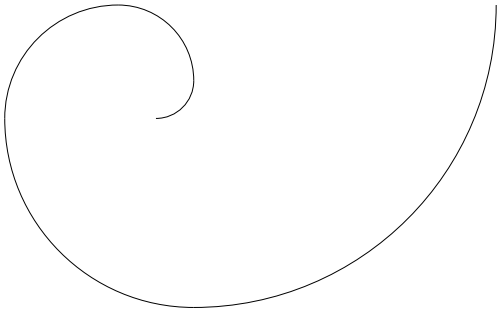
Turtle-Grafik

Erzeuge die Kurven.

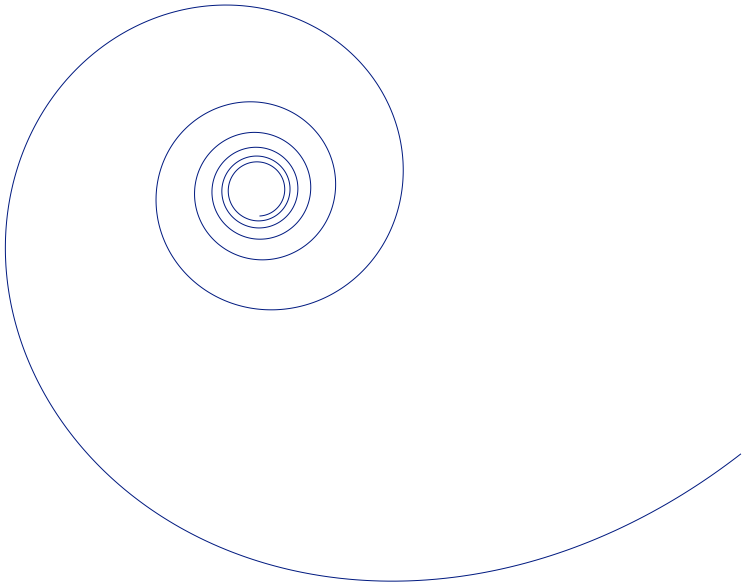
a)



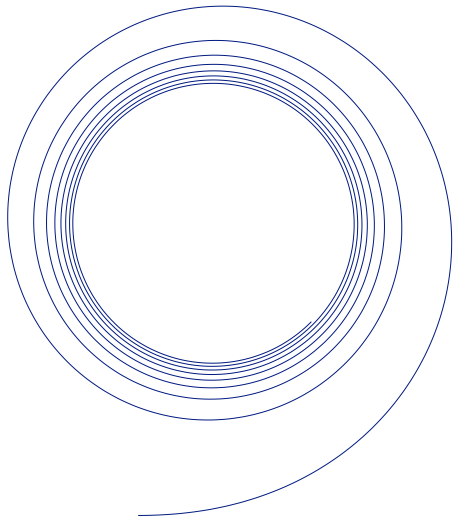
b)



c)



d)



Siehe auch: weitere Aufgaben
Startseite