CoderDojo Saar - Turtle
Aufgaben

Niklas Schneider - Maximilian Krahn

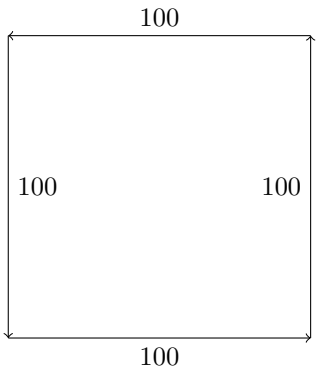
05.03. - 06.03.2021

Hinweis

Für jede Aufgabe gibt es eine eigene Datei, in der sich die Signaturen der zu vervollständigenden Funktionen befinden. Um einen Aufgabenteil auszuführen, rufe die entsprechende Funktion am Ende von `main.py` auf.

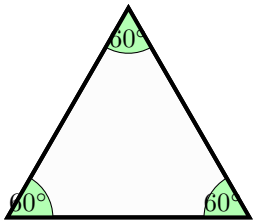
Aufgabe 1 - Die ersten Schritte

a) Zeichne ein Quadrat mit der Seitenlänge 100. Vervollständige dazu die Methode `quadrat()`.

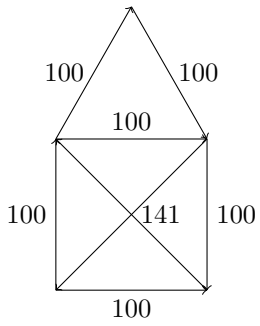


b) Zeichne ein gleichseitiges Dreieck. Vervollständige dazu die Methode `dreieck()`.

Zur Erinnerung: Bei einem gleichseitigen Dreieck sind alle Seiten gleich lang und jeder Innenwinkel hat 60°.



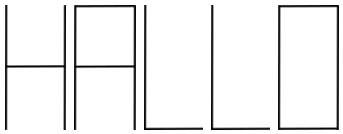
c) Zeichne das Haus des Nikolaus mit den gerade gelernten Befehlen. Vervollständige dazu die Methode `haus_des_nikolaus()`.



Aufgabe 2 - Buchstabensuppe

a) Zeichne die Buchstaben “MK” bunt. Vervollständige dazu die Methode `mk()`.

b) Zeichne deine eigenen Intialien bunt. Vervollständige dazu die Methode `initialien()`.

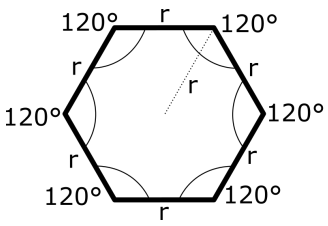


Aufgabe 3 - Verhexte Hexagone

a) Vervollständige die Funktion `hexagon(r)`, die ein Sechseck mit Radius `r` zeichnet.

Dabei soll die aktuelle Position der Turtle der Mittelpunkt des Sechsecks sein. Achte darauf, dass die Turtle wieder in dieselbe Richtung wie vorher schaut, wenn das Sechseck gezeichnet ist.

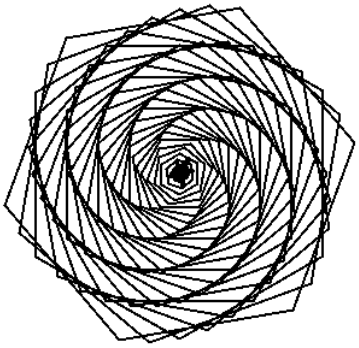
Ein Sechseck ist folgendermaßen aufgebaut:



b) Vervollständige nun die Funktion `magic_hex(n, alpha)`, sodass sie folgende Verhalten zeigt:

- *Parameter:* `n` - Anzahl Schritte, `alpha` - Drehwinkel.
- Die Funktion soll `n` Sechsecke zeichnen.
- Nach jedem Sechseck soll die Turtle um den Winkel `alpha` gedreht werden.
- Außerdem soll mit jedem Sechseck der Radius wachsen. Wähle dazu einen festen Startradius und erhöhe nach jedem Schleifendurchlauf den Radius.

Lass die Turtle das Bild malen und vergleiche es mit dem folgenden Bild.



c) Modifiziere nun die Funktion `hexagon` so, dass jede Seite des Sechsecks eine andere Farbe hat.

Benutze dazu die Liste `farben`. Du darfst die Farben auch ändern, weitere Farben findest du auf dem Cheat Sheet. Gegebenenfalls musst du ebenfalls die Hintergrundfarbe ändern, damit die Sechsecke erkennbar sind.

d) Spiele mit den Parametern und verändere die Funktionen nach deinen Wünschen. Fertige dabei Screenshots von deinen Ergebnissen an und erkläre, wie die Bilder zustande gekommen sind.

Aufgabe 4 - Das Wunder der Rekursion

a) Vervollständige die rekursive Funktion, `summe_rek(n)`, sodass sie folgendes Verhalten zeigt:

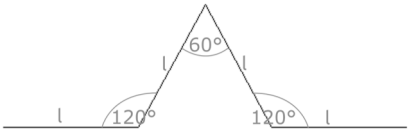
- *Parameter:* `n` - die Zahl, bis zu der aufsummiert wird.
- Es soll $1 + 2 + \cdots + n$ zurückgegeben werden.

b) Vervollständige die rekursive Funktion, `pot_rek(a, b)`, sodass sie folgendes Verhalten zeigt:

- *Parameter:* `a` - die Basis, `b` - der Exponent.
- Es soll $\underbrace{a \cdot a \cdot \cdots \cdot a}_{b \text{ mal}} = a^b$ zurückgegeben werden.

Aufgabe 5 - Schneeflocke

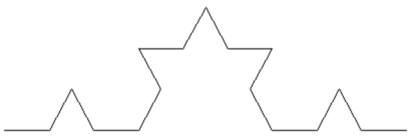
(a) Vervollständige die Funktion `koch_einfach(l)`, sodass sie mit der Turtle ausgehend von der aktuellen Position und Drehung das folgende Bild zeichnet.



(b) Vervollständige die Funktion `koch_rek(l, n)`, sodass sie das folgende Verhalten zeigt:

- *Parameter:* `l` - die Seitenlänge, `n` - die Anzahl der Schritte.
- Wenn nur noch ein Schritt übrig ist, soll die Funktion eine einfache Kochkurve der Länge l zeichnen.
- Ansonsten soll sie an den Stellen, an denen es geradeaus geht, stattdessen kleinere Kochkurven zeichnen. Die Anzahl an Schritten wird dabei um eins verkleinert.

So sieht das Ergebnis aus, wenn `koch_rek(l, n)` mit `n` $= 2$ aufgerufen wird:



(c) Eine Schneeflocke besteht aus mehreren aneinandergereihten Kochkurven in der Form eines n-Ecks.

Vervollständige die Funktion `schneeflocke(l, n, e)`, sodass sie das folgende Verhalten zeigt:

- *Parameter:* `l` - die Seitenlänge, `n` - die Anzahl der Schritte, `e` - die Anzahl der Ecken.
- Die Funktion soll ein n-Eck aus Kochkurven zeichnen. Ist zum Beispiel $n = 4$, soll sie ein Quadrat zeichnen, doch statt gerader Linien eine Kochkurve mit den gegebenen Parametern.

Finde die Anzahl an Ecken heraus, für die die Schneeflocke am schönsten aussieht.

Aufgabe 6 - Fibonacci und seine Zahlen

Die Fibonacci-Reihe ist eine besondere Zahlensequenz. Man findet immer die nächste Zahl, indem man die beiden vorhergehenden Zahlen zusammenaddiert. Die ersten beiden Zahlen sind 0 und 1. So ergibt sich:

$$\begin{aligned}\mathsf{fib}_0 &= 0\\ \mathsf{fib}_1 &= 1\\ \mathsf{fib}_2 &= 1\\ \mathsf{fib}_3 &= 2\\ \mathsf{fib}_4 &= 3\\ \mathsf{fib}_5 &= 5\\ \mathsf{fib}_6 &= 8\\ &\vdots\end{aligned}$$

Um die Aufgabe auszuführen, rufe in `main.py` die Funktion `Aufgabe6.zeichnen()` auf.

(a) Vervollständige die rekursive Funktion `fib_rec(n)`, sodass sie folgendes Verhalten zeigt:

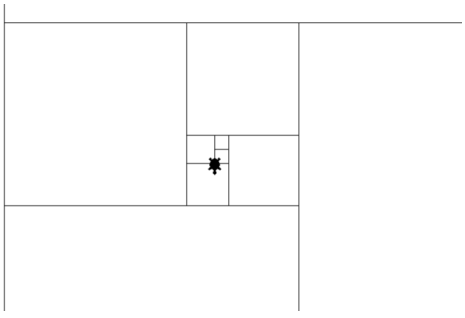
- *Parameter:* `n` - die Zahl, welche Fibonaccizahl gebildet wird.
- Es soll fib_n zurückgegeben werden.

(b) Vervollständige die Funktion `quatrat(l)`, die ein Quadrat mit der Seitenlänge `l` zeichnet.

(c) Vervollständige die Funktion `quadrate(n, l)` die das folgende Verhalten zeigt:

- *Parameter:* `n` - die Anzahl der Schritte, `l` - die Länge. Diese vergrößert das zu zeichnende Bild.
- Solange noch Schritte übrig sind, soll die Funktion im i-ten Schritt ein Quadrat mit der Seitenlänge $l \cdot \mathsf{fib}_i$ zeichnen.
- Die Quadrate sollen spiralförmig gegen den Uhrzeigersinn um den Startpunkt angeordnet sein.

Das Ergebnis sollte folgendermaßen aussehen:

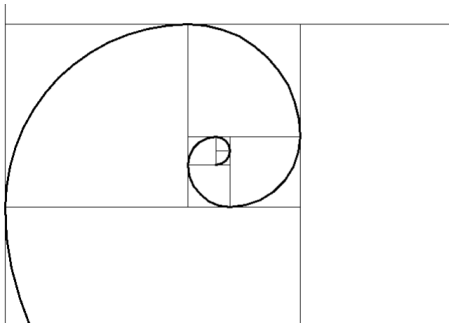


(d) Vervollständige die Funktion `kurve(n, s)`, welche die sogenannte Fibonacci-Kurve zeichnet. Gehe dazu wie folgt vor:

- *Parameter:* `n` - die Anzahl der Schritte, `l` - die Länge. Diese vergrößert das zu zeichnende Bild.

- Ein Kurvensegment besteht immer aus einem Viertelkreis. Ein solcher lässt sich mit `circle(r, 90)` zeichnen, wobei `r` der Radius des Kreises ist.
- Im Fall unserer Kurve ist der Radius des i-ten Kurvensegments genau die i-te Fibonacci-Zahl, also fib_i.

Das Ergebnis sollte folgendermaßen aussehen:

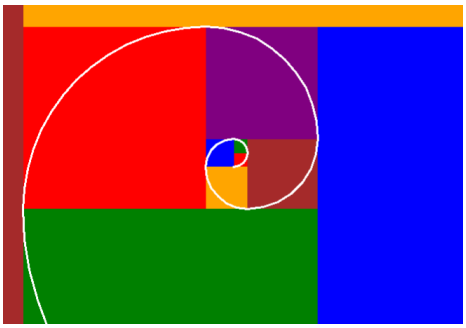


(e) Färbe die Quadrate bunt ein. Benutze dazu die Funktion `naechste_farbe(i)`, die als Parameter den aktuellen Schritt `i` annimmt.

Außerdem kannst du die Quadrate in der Funktion `quadrat(l)` farbig ausfüllen. Benutze dazu die Turtle-Funktionen `begin_fill()` und `end_fill()`.

Denke daran, auch danach noch die Kurve einzufärben, damit man sie auf dem bunten Hintergrund noch erkennt.

Das Ergebnis sollte folgendermaßen aussehen:



Du kannst die Liste `farben` nach Belieben ergänzen oder verändern. Experimentiere außerdem mit verschiedenen Längen und mache Screenshots von deinen Resulaten, die du präsentieren kannst.