Programmieren mit Jython – Teil 6: Rekursion

Lernziele

- Verstehen, was eine Rekursion ist, und wie man rekursiv programmiert.

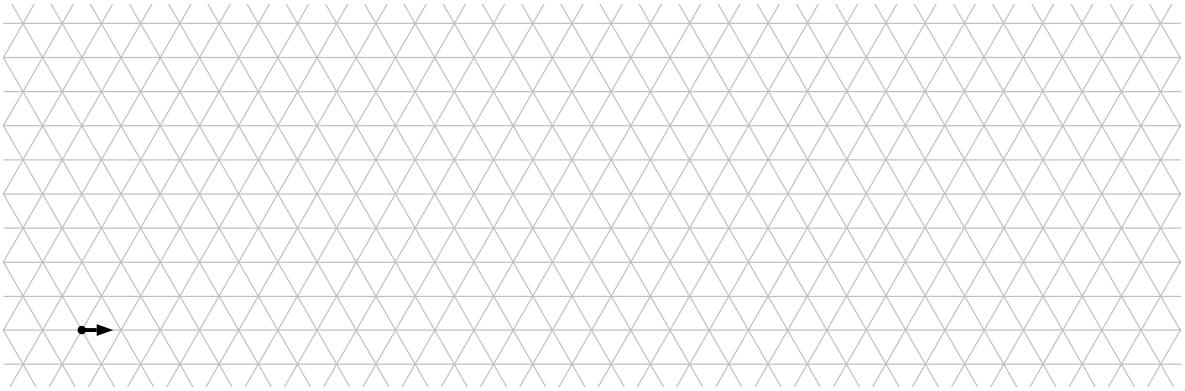
Erstes Beispiel einer Rekursion

Betrachte das Unterprogramm `KochZacken(laenge, Stufe)`. Das Spezielle daran ist, dass innerhalb des Unterprogramms das Unterprogramm selber aufgerufen wird:

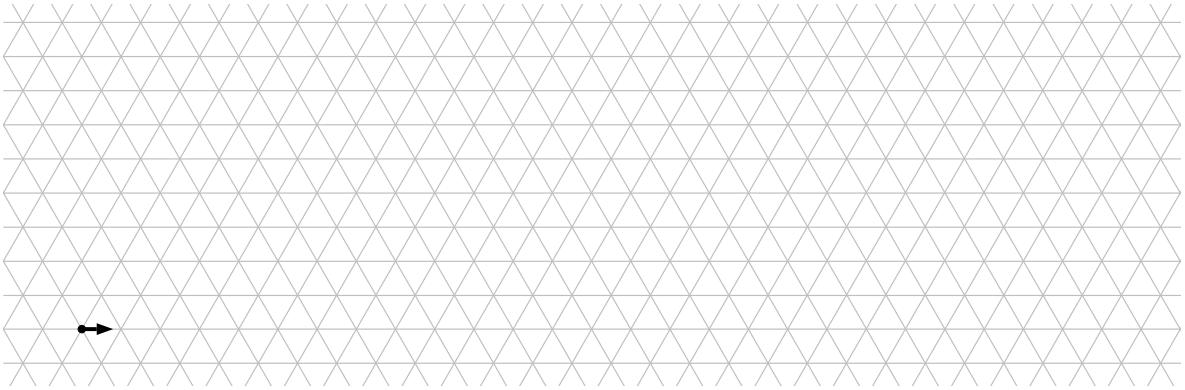
```
def KochZacken(laenge, Stufe):
    if Stufe>0:
        KochZacken(laenge/3, Stufe -1)
        left(60)
        KochZacken(laenge/3, Stufe -1)
        right(120)
        KochZacken(laenge/3, Stufe -1)
        left(60)
        KochZacken(laenge/3, Stufe -1)
    else:
        forward(laenge)
```

Aufgaben: Koch-Schneeflocke

6.1) Im folgenden Gitter beträgt der Abstand zwischen benachbarten Gitterpunkten 10 Einheiten. Zu Beginn ist die Turtle im markierten Punkt und schaut nach rechts. Zeichne den Weg, den die Turtle beim Aufruf `KochZacken(270,1)` zurücklegt:

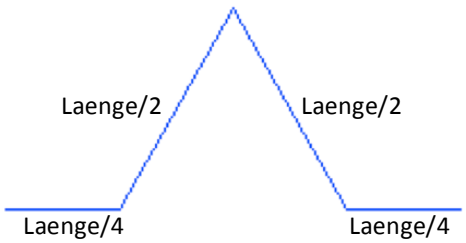


6.2) Zeichne den von der Turtle zurückgelegten Weg, wenn `KochZacken(270,3)` aufgerufen wird:



6.3) Ändere das Programm `KochZacken` geeignet ab, sodass jedes Segment jeweils durch einen Streckenzug ersetzt wird, der dem nebenstehenden ähnlich ist. Benenne das Programm neu

`GrossZacken(Laenge, Stufe).`



Aufgaben: Zweige

Ziel dieser Serie von Aufgaben ist es, ein Programm zu schreiben, das mehr oder weniger realistische Zweige zeichnen kann. In einem weiteren Schritt soll danach der ``Realitätsgrad'' erhöht werden.

6.4) a) Definiere zuerst vier Variablen wie folgt:

```
from gturtle import *
makeTurtle()
winkelL=30
winkelR=35
faktorL=0.5
faktorR=0.6
```

b) Schreibe nun ein Programm `Zweig(laenge,rek)`, das eine der zwei folgenden Figuren zeichnet, je nach dem ob der Wert der Variable `rek` grösser als Null ist oder nicht.

Wichtig ist, dass die Turtle zu Beginn und nach Ablauf unten ist und nach oben schaut. Verwende dazu den Befehl `back`.

Die vertikale Strecke hat die Länge `laenge`. Ist `rek>0` so soll der links abstehende Ast durch den Winkel `winkelL` aus der Vertikalen gedreht sein und die Länge `laenge*faktorL` haben. Analog soll der rechte Ast um den Winkel `winkelR` nach rechts aus der Vertikalen gedreht sein und die Länge `laenge*faktorR` haben.

c) Ersetze nun im Unterprogramm `Zweig` die folgenden vier Codezeilen:

Vorher	Ersetzt durch
`forward(laenge*faktorL)`	`Zweig(laenge*faktorL,rek-1)`
`back(laenge*faktorL)`	[weglassen]
`forward(laenge*faktorR)`	`Zweig(laenge*faktorR,rek-1)`
`back(laenge*faktorR)`	[weglassen]

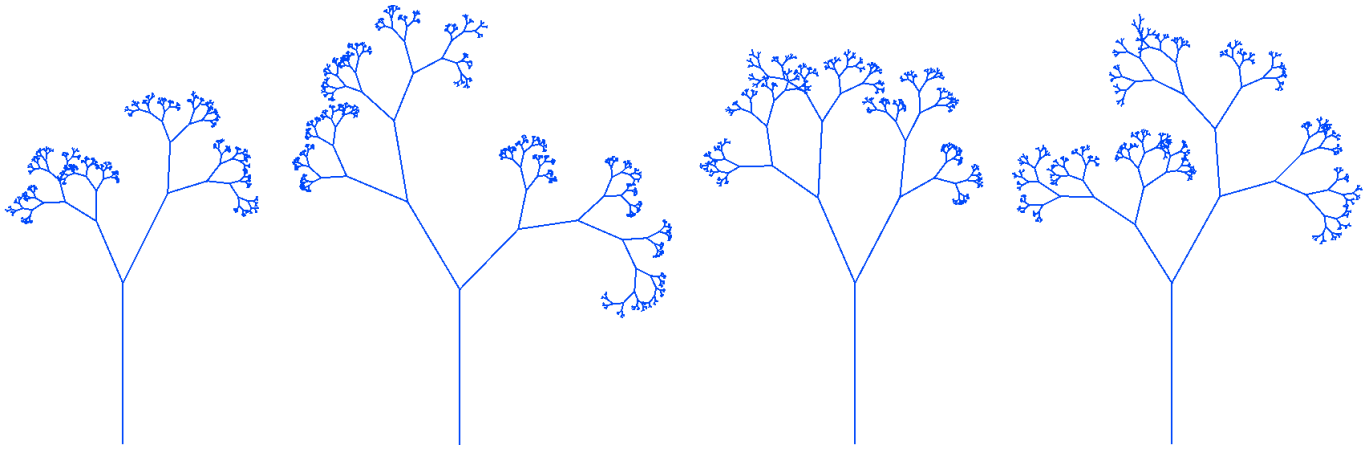
d) Teste das Programm aus durch:

```
hideTurtle()
setPos(0,-280)
Zweig(200,5)
```

e) Erzeuge verschiedene Zweige indem Du die Werte der Variablen zu Beginn veränderst.

Zusatzaufgaben:

Realistischere Zweige erhält man, wenn man etwas Zufall ins Spiel bringt. So wurden die folgenden Bilder erzeugt, und zwar alle mit denselben Werten der Variablen (die Unterschiede entstehen aus den verschiedenen zufälligen Zahlen, die im Prozess erzeugt werden).



6.5) Um die Werte zufällig leicht abzuändern soll folgende Funktion verwendet werden: `ZufallsFaktor(r)`. Eine Funktion ist ein Programm, das einen Wert zurückgeben kann. Die Rückgabe erfolgt mit dem Befehl **`return`**`(wert)`.

```
from random import *
def ZufallsFaktor(r):
    wert = uniform(1-r,1+r)
    return(wert)
```

Die Funktion `uniform(a,b)` liefert eine zufällige reelle Zahl zwischen `a` und `b`. Sie ist Teil der Bibliothek random, die zu Beginn geladen werden muss.

6.6) Füge nun noch zwei weitere Variablen zu den bestehenden 4 hinzu:

```
paramL=0.3
paramR=0.4
```

und ändere das bestehende Programm ab: sowohl der Winkel, als auch die Länge soll in jedem Schritt zufällig abgeändert werden. Definiere dazu Variablen:

```
zufallWinkel=winkelL*ZufallsFaktor(paramL)
zufallLaenge=laenge*faktorL*ZufallsFaktor(paramL)
```

bzw.

```
zufallWinkel=winkelR*ZufallsFaktor(paramR)
zufallLaenge=laenge*faktorR*ZufallsFaktor(paramR)
```

und verwende diese Werte für die Drehung und die Länge.

6.7) Variiere die sechs Variablen `winkelL`, `winkelR`, `faktorL`, `faktorR`, `paramL`, `paramR` um immer neue Zweige zu kreieren.

Anwendung

Solche, durch Rekursion entstandene und mit dem Zufall leicht deformierte, Figuren werden in der Animation von Zeichentrickfilmen routinemässig angewandt, um möglichst realistische Bäume, Sträucher, Berge, Wolken und vieles mehr zu erzeugen. Ist die Form einmal erzeugt, werden die Gebilde dann mit geeigneten Texturen überzogen.

Hausaufgaben

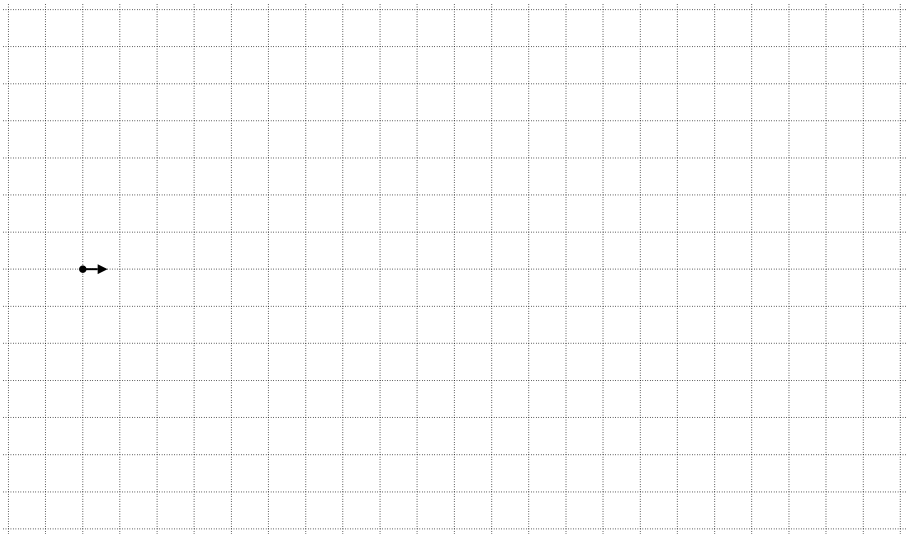
Löse folgende Aufgaben mit **Papier und Bleistift** (ohne Computer):

H6.1) Das folgende Programm ist rekursiv programmiert.

```
def Figur(l,rek):
    forward(l)
    if rek>0:
        left(90)
        Figur(l/2,rek-1)
        right(180)
        Figur(l/2,rek-1)
        left(90)
    back(l)
```

Zeichne in den folgenden Bildern die Figuren, welche die angegebenen Programmaufrufe erzeugen. Die Häuschen sind als Referenz angegeben, jedes Häuschen hat die Seitenlänge 10. Zu Beginn ist die Turtle an der Position des schwarzen Punktes und schaut in Richtung des Pfeiles.

a) `Figur(120,0)`



b) `Figur(120,3)`