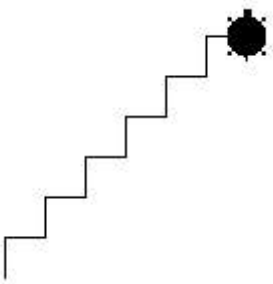
15. REKURSIONEN

DU LERNST HIER...

dass Rekursion ein Lösungsverfahren ist, bei dem ein Problem auf das gleichartige, aber etwas vereinfachte Problem zurückgeführt wird. Eine Funktion ist dann rekursiv, wenn sie sich selbst aufruft. Damit sich eine rekursive Funktion nicht endlos aufruft, braucht sie eine Abbruchbedingung. Betrachte dieses Kapitel als Zusatzstoff und freue dich an schönen Bildern, die mit Rekursionen entstehen.

MUSTERBEISPIELE

Das Prinzip einer Rekursion kann anschaulich am Beispiel des Zeichnens einer Treppe erklärt werden: Eine Treppe mit 6 Stufen wird aus einer Stufe und einer Treppe mit 5 Stufen zusammengesetzt, ein Treppe mit 5 Stufen wird auf eine Stufe und einer Treppe mit 4 Stufen zurückgeführt (usw.). In der Funktion *staircase(n)* wird also jeweils dieselbe Funktion *staircase(n - 1)* aufgerufen. Wenn die Abbruchbedingung n = 0 erfüllt ist, bricht das Programm mit *return* ab.

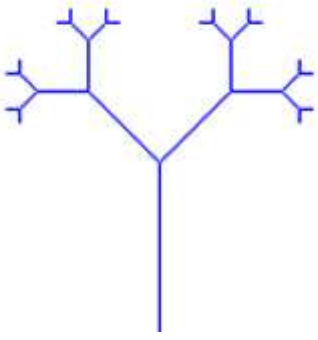


Programm: **[► Online-Editor]** **[► WebTigerJython]**

```
from gturtle import *

def step():
    forward(20)
    right(90)
    forward(20)
    left(90)

def staircase(n):
    if n == 0:
        return
    step()
    staircase(n-1)

makeTurtle()
staircase(6)
```

► In Zwischenablage kopieren

Das Zeichnen eines binären Baums lässt sich auf eine einfache Grundfigur zurückführen.



Anschaulich gesagt ist ein binärer Baum mit der Stammlänge s ein Stamm und ein Links- sowie ein Rechtsbaum mit der Stammlänge s /2. Für die Stammlänge < 8 bricht die Rekursion ab.

Programm: **[► Online-Editor]** **[► WebTigerJython]**

```
from gturtle import *

def tree(s):
    if s < 8:
        return
    forward(s)
    left(45)
    tree(s/2)
    right(90)
    tree(s/2)
    left(45)
    back(s)

makeTurtle()
setY(-100)
tree(128)
```

► In Zwischenablage kopieren

Bemerkung: das back(s) ist wichtig, da nach dem Zeichnen des (Teil-)Baums die Turtle wieder im Anfangszustand sein muss.

MERKE DIR...

Die Rekursion ist ein wichtiges Lösungsverfahren, bei dem man ein Problem auf dasselbe, etwas vereinfachte Problem zurückführt und dieses in einem besonders einfachen Fall löst (bei Rekursionsabbruch). Rekursive Lösungsverfahren sind elegant, aber meist schwierig zu durchblicken.

WEITERE BEKANNTE REKURSIONEN

Sirpinsky Fractal **[► Online-Editor]** **[► WebTigerJython]**

Flower Fractal **[► Online-Editor]** **[► WebTigerJython]**

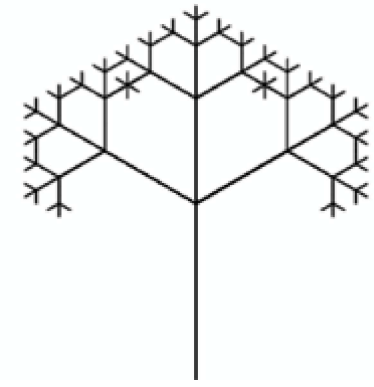
Peano Fractal **[► Online-Editor]** **[► WebTigerJython]**

Tree Fractal **[► Online-Editor]** **[► WebTigerJython]**

Koch Fractal **[► Online-Editor]** **[► WebTigerJython]**

ZUM SELBST LÖSEN

1. Überlege, wie du das Programm, das den binären Baum zeichnet (Beispiel 2) ändern musst, damit der nebenstehende Baum entsteht.



2. Ergänze den Programmcode so, dass eine Flocke gezeichnet wird. Gehe von folgendem Gerüst aus.

```
def figure(s):

    for i in range(6):
        forward(s)
        figure(s/3)
        back(s)
        right(60)
```