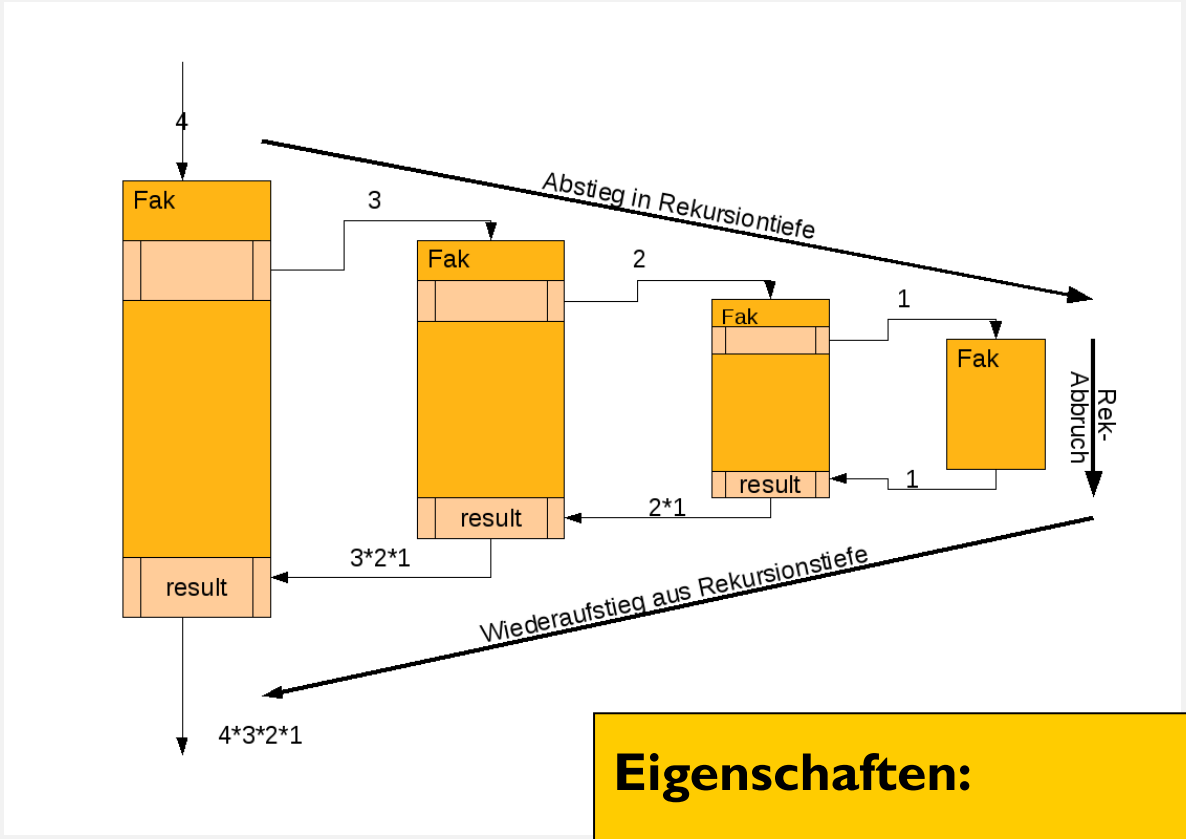
WIEDERHOLUNG REKURSION

- Video

EIGENSCHAFTEN REKURSIVER FUNKTIONEN




```
def fak(n):
    if n==1:
        return 1
    else:
        return n*fak(n-1)

print(fak(10))
```

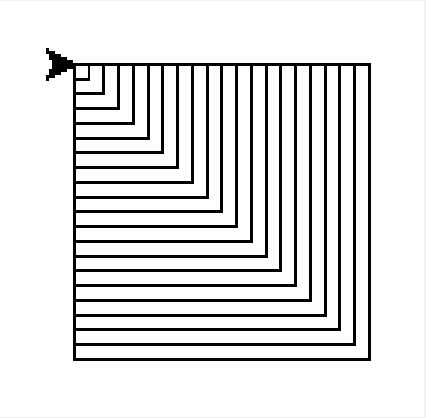
Eigenschaften:

fak(n)=n * fak(n-1) (Selbstaufruf mit geänd. Parametern)

fak(1) = 1 (Abbruchbedingung)

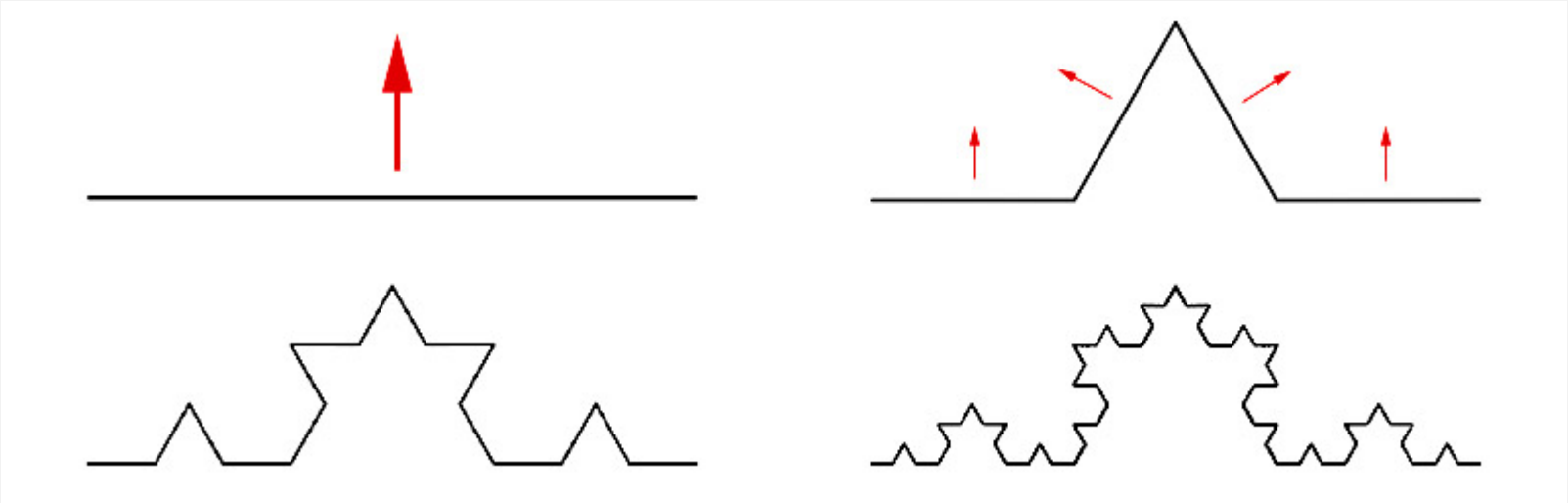
WIR KÖNNEN

- Fakultät
- Summen der ersten n Zahlen rekursiv:
  - sum(n)=n+sum(n-1)
  - sum(1)=1
- Einfache rekursive Grafiken



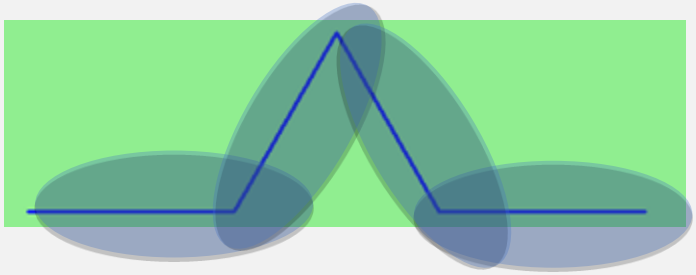
REKURSIVE GRAFIKEN

- Gegeben sei die folgende Konstruktionsanweisung:
- Zeichne eine Strecke („Initiator“)
- Zerlege die Strecke in drei gleich große Teile.
- Errichte über der mittleren Strecke ein gleichseitiges Dreieck.
- Entferne die Grundseite des Dreiecks.
- Es entstehen 4 Strecken. Führe mit allen die Schritte 1-5 aus.

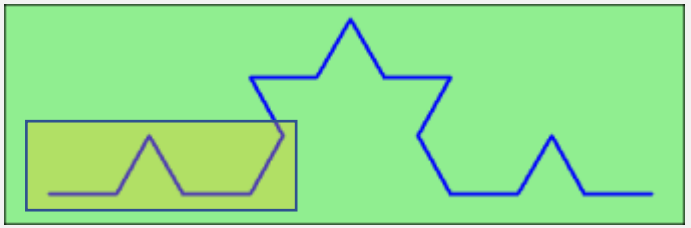


DENKEN WIE EIN „REKURSIONIST“

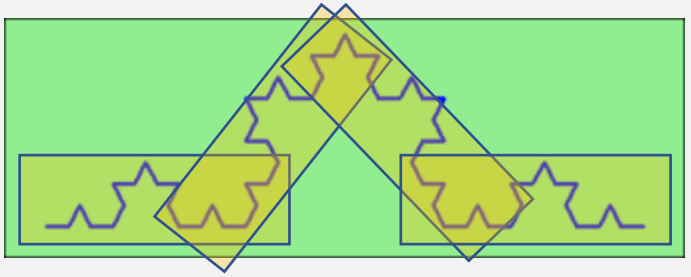
Stufe 0



Stufe 1 Für Stufe 1 benötigen wir 4 Stufe 0 Fraktale



Stufe 2 Für Stufe 2 benötigen wir 4 Stufe 1 Fraktale

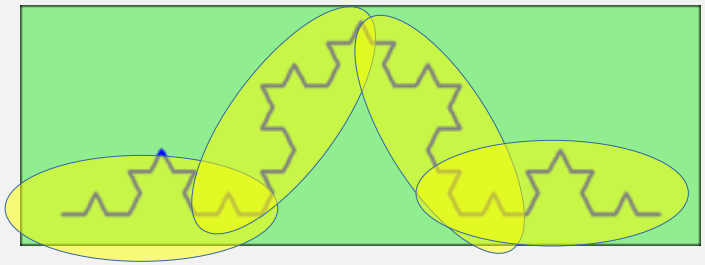


Stufe 3 Für Stufe 3 benötigen wir 4 Stufe 2 Fraktale

Gezeichnet wird nur in Stufe 0!

ALGORITHMUS -IDEE:

- Algorithmus für Koch Kurve der Stufe n:
  - Zeichne Koch Kurve der Stufe n-1
  - Drehe links um 60°
  - Zeichne Kurve der Stufe n-1
  - Drehe rechts um 120°
  - Zeichne Kurve der Stufe n-1
  - Drehe links um 60°
  - Zeichne Kurve der Stufe n-1
  - (Abbruch bei Stufe 0, dann Zeichnen der Geraden)



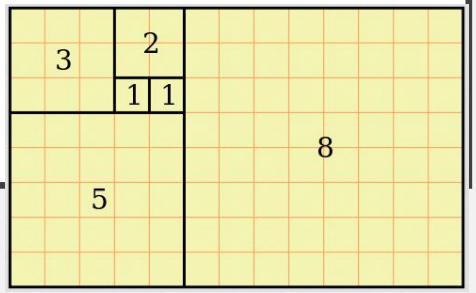
ALGORITHMUS KOCH KURVE

```
from turtle import *

def koch(stufe, groesse):
    if stufe == 0:
        fd(groesse)
    else:
        koch(stufe-1, groesse)
        left(60)
        koch(stufe-1, groesse)
        right(120)
        koch(stufe-1, groesse)
        left(60)
        koch(stufe-1, groesse)

penup() #Stift positionieren
goto(-200,0)
pendown()
koch(3, 100) # Aufruf
mainloop()
```

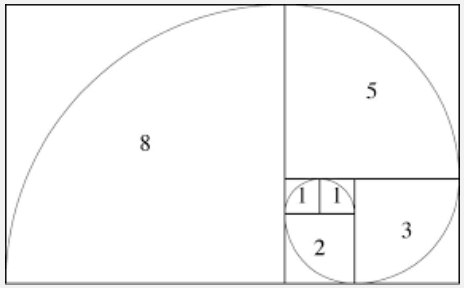
Aufgabe:

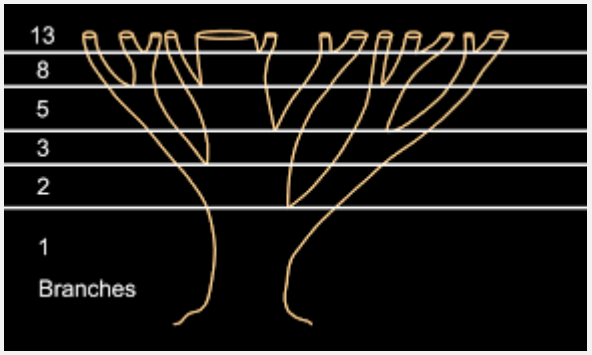
1. Programmiere die Koch Kurve. Teste mit verschiedenen Tiefen und Größen.
2. Weise die Eigenschaften der Rekursion am Quelltext nach.

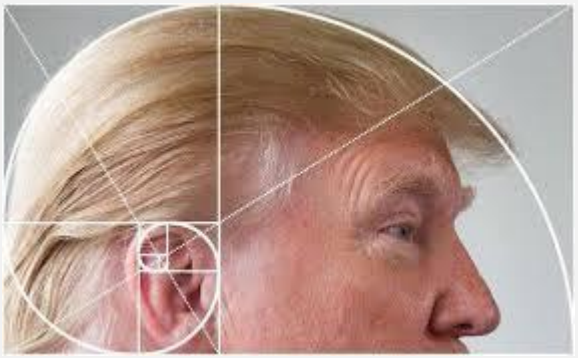
FIBONACCI ZAHLEN



- Die ist eine unendliche Folge natürlicher Zahlen, die in der Natur sehr häufig anzufinden ist. Ein Glied der Folge entsteht jeweils als Summe der vorhergehenden Folgeglieder. Die Startglieder sind 1,1.
- Ergebnis: 1,1,2,3,5,8,13 …
- Schreibe eine rekursive Funktion fib(n), die jeweils die ersten n Zahlen der Fibonacci Folge ausgibt.







FIBONACCI ZAHLEN - LÖSUNG

```
def fibListe(n):
    if n==1 or n==2:
        return 1
    else:
        Liste=[1,1]
        for i in range(2,n):
            val=Liste[i-1]+Liste[i-2]
            Liste.append(val)
    return Liste

def fib(n):
    if n==1 or n==2:
        return 1
    else:
        return fib(n-1)+fib(n-2)
```

Iterative Lösung

Rekursive Lösung

FIBONACCI ZAHLEN - MEMOIZATION

```
fibCache={}
def fibSpeicher(n):
    if n==1 or n==2:
        val=1
    else:
        val=fibSpeicher(n-1)+fibSpeicher(n-2)
    fibCache[n]=val

    return val
```

Lösung mit
„Funktionsspeicher“
-
Cache

WEITERE AUFGABEN

- -> Kopie