7. PARAMETER

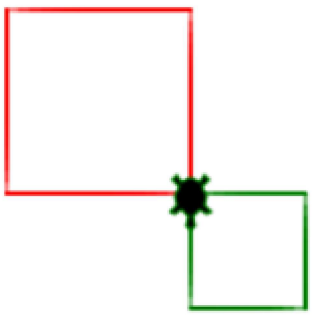
DU LERNST HIER...

wie du Funktionen mit Parametern definieren kannst. Du kennst Parameter bereits von vielen Turtlebefehlen. Beim Befehl *forward(s)* darfst du für s verschiedene Zahlen einsetzen. Mit *forward(100)* bewegt sich Turtle 100 Schritte vorwärts. Der Befehl *forward(s)* hat einen Paramter s. Auch selbst definierte Funktionen können Parameter haben. O►

MUSTERBEISPIELE

Im Kapitel 5 hast du eine Funktion *square()* definiert, die ein Quadrat mit fixer Seitenlänge 100 zeichnet. Man sagt anschaulich, dass die Seitenlänge 100 im Programm "*fest verdrahtet*" sei.

Die Funktion kann viel flexibler eingesetzt werden, wenn du die Seitenlänge beim Funktionsaufruf angeben kannst, also z.B.*square(50)* oder *square(70)* schreiben kannst. Dazu musst du die Funktionsdefinition von **square(s)** mit einem Parameter versehen, dessen Name du in die Parameterklammer schreibst. Den Parameter kannst du im Innern der Funktion (im Funktionskörper) wie eine gewöhnliche Variable verwenden. Im Programm zeichnet die Turtle zwei Quadrate mit den Seitenlängen 80 und 50.



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

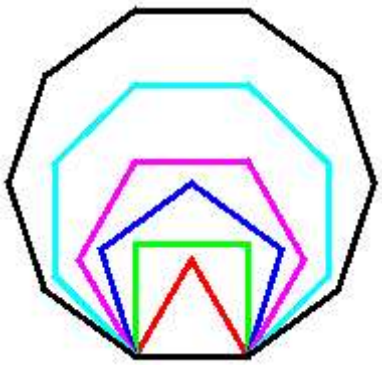
```
from gturtle import *

def square(s):
     repeat 4:
        forward(s)
        left(90)

makeTurtle()
setPenColor("red")
square(80)
left(180)
setPenColor("green")
square(50)
```

► In Zwischenablage kopieren

Eine Funktion kann auch mehrere Parameter haben. In deinem Beispel definierst du eine Funktion **polygon(n, c)**mit zwei Parametern: n für die Anzahl Ecken, c für die Stiftfarbe. Die Funktion zeichnet regelmässige Vielecke mit der gegebenen Anzahl Ecken und Farbe. Im Hauptprogramm wird die Funktion 6 mal aufgerufen und zeichnet dabei ein Dreieck, Quadrat, 5-Eck, 6-Eck, 8-Eck und 10 Eck un verschiedenen Farben.



Der Drewinkel, den du zum Zeichnen benötigst, lässt sich mit **360 / n** berechnen.

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

```
from gturtle import *

def polygon(n, c):
    w = 360 / n
    setPenColor(c)
    repeat n:
        forward(100)
        left(w)

makeTurtle()
setPos(-50, -200)
setPenWidth(3)
right(90)
polygon(3, "red")
polygon(4, "green")
polygon(5, "blue")
polygon(6, "magenta")
polygon(8, "cyan")
polygon(10, "black")
```

► In Zwischenablage kopieren

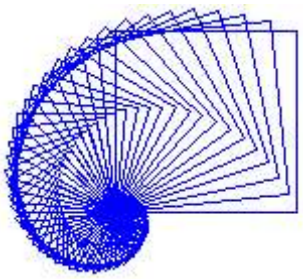
MERKE DIR...

Wenn du die Funktion *square(s)* mit den Parametern 100 aufrufst, erhält die Variable *s* in der Funktion *square(s)* Wert 100. Die Parametrisierung von Funktionen ist wichtig, denn dadurch kannst du die Funktionen flexibler verwenden. O►

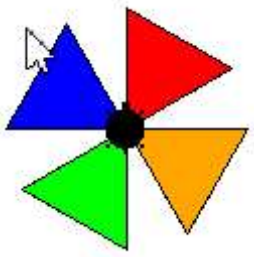
Mit der Funktion **setPos(x, y)** kannst du die Turtle an eine beliebige Position im Turtlefenster versetzen.

ZUM SELBST LÖSEN

1. Das nebenstehende Bild entsteht, indem die Länge der Quadratseite ausgehend von 180 bei jedem nachfolgenden Quadrat mit dem Faktor 0.95 verkleinert wird. Es werden 100 Quadrate gezeichnet. Schreibe das Programm mit der Funktion *square(s)*



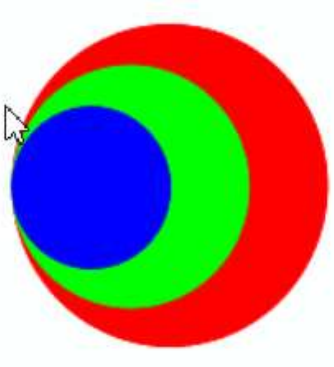
2. Definiere eine Funktion, *dreieck(c)*, welche ein gefülltes Dreieck mit der gegebenen Farbe zeichnet. Erstelle damit das nebenstehende Bild.



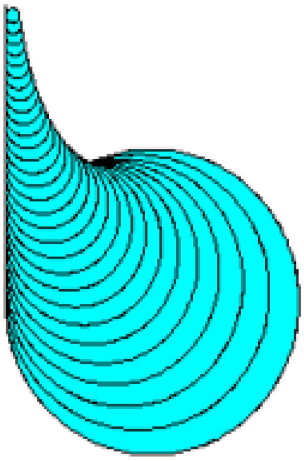
3. Definiere eine Funktion *circle(s, c)*, mit dem die Turtle gefüllte Kreise mit gegebener Füllfarbe zeichnet und erstelle damit die nebenstehende Zeichnung. Du hast bereits im

Kapitel *Figuren füllen* gelernt, dass man einen Kreis als ein Vieleck zeichnen kann.

```
repeat 120:
    forward(3)
    right(3)
```



4. Du definiert eine Funktion circle(s), die einen Kreis zeichnet. Die nebenstehende Figur entsteht, indem du 30 Kreise zeichnest, s nach jedem gezeichneten Kreis verkleinerst (s = s *0.9) und die Turtle 5 Schritte vorwärts bewegst.



5. Definiere eine Funktion square(s, c) mit zwei Parametern: s für die Seitenlänge und c für die Füllfarbe. Rufe dann die Funktion dreimal auf, so dass die nebenstehende Figur entsteht.