2. Wiederholungen

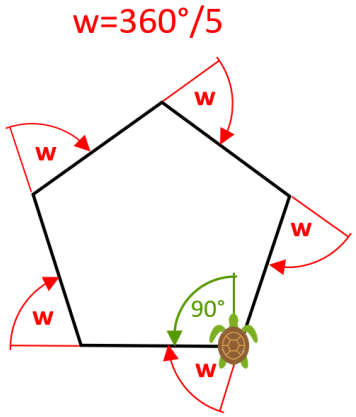
Lernziele:

- Du lernst, wie man wiederholende Programmteile abgekürzt darstellen kann.
- Du lernst, dass man durch das Erkennen von repetitiven Mustern ein Verständnis für den Aufbau von gewissen Strukturen gewinnen kann.

In der Informatik ist das Erkennen und Beschreiben von **Mustern** eine wichtige Tätigkeit, die dir auch beim Entwickeln von einfacher lesbaren Programmen hilft.

2.1 Einfache Schleifen

In Aufgabe 1.4 hast du ein Programm für ein regelmässiges Fünfeck entwickelt. Die Zeilen 7 bis 16 wiederholen sich:




```
1  from gturtle import*
2
3  makeTurtle()
4
5  lt(90)
6
7  fd(80)
8  rt(360/5)
9  fd(80)
10 rt(360/5)
11 fd(80)
12 rt(360/5)
13 fd(80)
14 rt(360/5)
15 fd(80)
16 rt(360/5)
```

In TigerJython kann man mit der Anweisung `repeat n`: einen Programmblock `n`-mal wiederholen lassen. Der zu wiederholende Programmblock muss eingerückt sein.

```
1 from gturtle import*
2
3 makeTurtle()
4
5 lt(90)
6
7 repeat 5:
8     fd(80)
9     rt(360/5)
```

Doppelpunkt

Körper der Schleife (eingerückt)

Begriffe

`repeat anzahl`: ist eine strukturierte Anweisung, die einen eingerückten Programmblock `anzahl`-mal wiederholt. Die Programmstruktur nennen wir **Schleife**. Der Anweisung muss immer ein Doppelpunkt folgen und der zu wiederholende Programmblock muss eingerückt sein.

Aufgabe 2.1

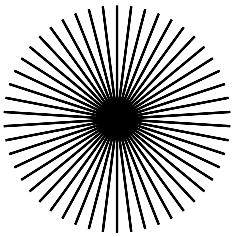
Schreibe ein Programm, das ein regelmässiges 12-Eck mit Seitenlänge 50 zeichnet.

Tipp:
Mit Hilfe des Befehls `hideTurtle()` bzw. `ht()` kann die Turtle ausgeblendet werden. Das Zeichnen wird dann massiv beschleunigt:

```
1 from gturtle import *
2 makeTurtle()
3
4 ht()  #Blendet die Turtle aus und beschleunigt das Zeichnen.
```

Aufgabe 2.2

Schreibe ein Programm, welche die abgebildete Figur mit 50 Strahlen der Länge 100 zeichnet:



Aufgabe 2.3

```
1 from gturtle import *
2 makeTurtle()
3
4 repeat 4:
5     fd(50)
6     lt(90)
7     fd(200)
```

(a)

```
1 from gturtle import*
2
3 makeTurtle()
4
5 repeat 3:
6     rt(60)
7     fd(100)
```

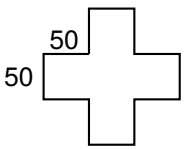
(b)

Programm (a): Die Ausgabe sollte ein Quadrat mit Seitenlänge 150 sein, es ist aber etwas falsch. Ändere **einen** der drei Parameter so, dass die Ausgabe korrekt ist!

Programm (b): Was ist die Ausgabe des Programms?

Aufgabe 2.4

Schreibe ein Programm, welches das abgebildete Kreuz zeichnet. Das Programm soll so wenig Zeilen haben wie möglich.

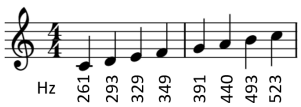


Aufgabe 2.5

a) Schreibe ein Programm, das einen Kreis zeichnet.

b) Ändere dein Programm so ab, dass der Kreis den Radius 100 hat. (Tipp: $\pi \approx 3.1415$)

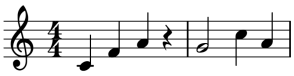
Rechts sind die Frequenzen der Oktavtöne von c' bis c" abgebildet. Mit dem Befehl `playTone(freq,t)` wird ein Ton mit der Frequenz **`freq`** in Hertz (Hz) und der Tonlänge **`t`** Millisekunden (ms) abgespielt. Für eine Viertelnote verwenden wir die Tonlänge 500 ms.



Beispiel 2.1

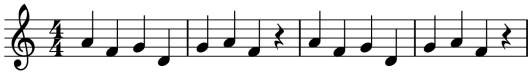
```
1  from gturtle import *
2  makeTurtle()
3
4  playTone(261,500)
5  playTone(349,500)
6  playTone(440,500)
7  delay(500)
8  playTone(391,1000)
9  playTone(523,500)
10 playTone(440,500)
```

Das Programm (links) spielt die Tonfolge (rechts) ab. Mit dem Befehl `delay(t)` kann eine Pause von **`t`** Millisekunden eingefügt werden.



Aufgabe 2.6

Schreibe ein möglichst kurzes Programm, das die Melodie spielt:



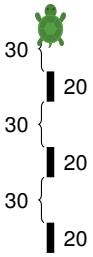
2.2 Stiftbreite und Zeichenmodus

Mit den folgenden Befehlen lassen sich Stiftbreite und Zeichenmodus festlegen:

`penUp()`	Hebt den Stift (Zeichnen wird deaktiviert)	`pu()`
`penDown()`	Senkt den Stift (Zeichnen wird aktiviert)	`pd()`
`setPenWidth(breite)`	Definiert die Stiftbreite	

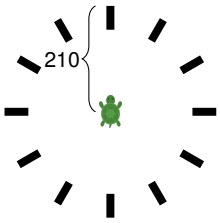
Beispiel 2.2

```
1  from gturtle import *
2  makeTurtle()
3
4  setPenWidth(5)
5  repeat 3:
6      forward(20)
7      pu()
8      forward(30)
9      pd()
```



Aufgabe 2.7

a) Schreibe ein Programm, das ein Zifferblatt mit einem Durchmesser von 420 zeichnet. Wähle für die Stundenstriche die Länge=30 und die Breite=15.



b) Ergänze das Programm mit einer zweiten Schleife, welche die Minutenstriche zeichnet. Wähle für die Minutenstriche die Länge=10 und Breite=3.

2.3 Stiftfarbe wählen

Mit dem folgenden Befehl lässt sich die Stiftfarbe festlegen:

`setPenColor(Farbe)`	Die Farbe muss immer in Hochkommas angeben werden: z.B. `"blue"`

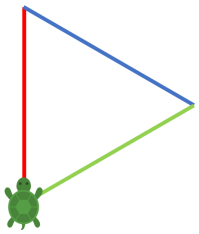
Für die Farben kannst du eine Auswahl von X11-Farbnamen verwenden:

yellow	red	navy	green	white
gold	magenta	blue	dark green	gray
orange	purple	cyan	sienna	black

Die X11-Palette ist eine Sammlung von RGB-Farben, denen Farbnamen zur einfachen Verwendung zugewiesen sind. Sie wurde vom MIT (Massachusetts Institute of Technology) entwickelt.

Beispiel 2.3

```
1  from gturtle import*
2  makeTurtle()
3
4  setPenColor("red")
5  fd(100)
6  rt(120)
7  setPenColor("blue")
8  fd(100)
9  rt(120)
10 setPenColor("green")
11 fd(100)
12 rt(120)
```



2.4 Flächen färben

Mit den folgenden Befehlen kannst du die Füllfarbe einer geschlossenen Figur festlegen:

`setFillColor(Farbe)`	Legt die Füllfarbe einer geschlossenen Figur fest.
`startPath()` und `fillPath()`	Markiert den Beginn und das Ende der zu füllenden Figur

Beispiel 2.4

```
1  from gturtle import*
2
3  makeTurtle()
4
5  setFillColor("orange")
6  setPenColor("red")
7
8  startPath()
9  repeat 4:
10     fd(50)
11     rt(90)
12 fillPath()
```

Füllfarbe

Farbe des Stifts

Start der Färbung

Figur, die gefärbt werden soll

Ende der Färbung

Aufgabe 2.8

Färbe den Flächeninhalt des Sterns aus Aufgabe 1.5 mit der Stift- und Füllfarbe `"yellow"`:

2.5 Kurzer Exkurs zu Datentypen

Dem Befehl `setPenColor("red")` übergeben wir als Parameter eine Farbe in Anführungszeichen. Warum? Daten werden im Computer als Folge von 0 und 1 gespeichert. Damit diese korrekt interpretiert und verarbeitet werden können, muss klar sein, um was für eine Art von Daten es sich handelt. Sobald Zeichen, Worte oder Sätze in Anführungszeichen stehen, weiss der Computer, dass es sich um **Text** handelt und nicht um Zahlen (mit denen man z.B. rechnen kann).

Begriffe

Mit **Datentypen** wird die Art von Daten definiert, damit der Computer diese korrekt interpretieren kann. Wir verwenden momentan drei grundlegende Datentypen:

1. Mit **String** (`str`) bezeichnen wir Buchstaben/Zeichen oder Text in Hochkommas.
   Beispiele: `"a"` oder `"Ganze Saetze sind auch moeglich"`
2. Mit **Integer** (`int`) bezeichnet ganze Zahlen wie z.B. 4 oder 500
3. Mit **Float** (`float`) bezeichnen wir Kommazahlen wie z.B. 0.5 oder 1.432

2.6 Verschachtelte Schleifen

Oft treten Wiederholungen in Wiederholungen auf, was man **verschachtelte Schleifen** nennt.

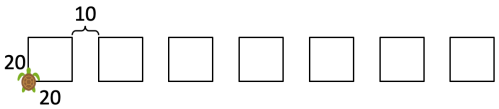
Beispiel 2.5

Im abgebildeten Programm wird mit der inneren Schleife ein Quadrat gezeichnet. Dieses wird mit der äusseren Schleife 7-mal in einem Abstand von 10 wiederholt gezeichnet.

```
1  from gturtle import *
2  makeTurtle()
3  ht()
4
5  repeat 7:
6      repeat 4:
7          fd(20)
8          rt(90)
9      pu()
10     rt(90)
11     fd(30)
12     lt(90)
13     pd()
```

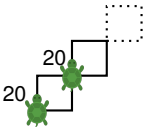
Quadrat

Verschiebung



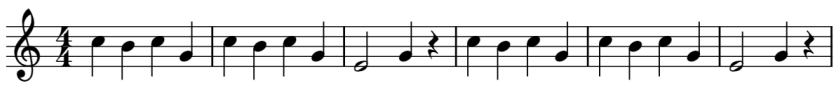
Aufgabe 2.9

Erstelle die Treppe aus 12 Quadraten mit Hilfe einer verschachtelten Schleife:

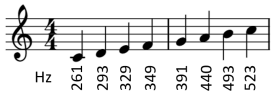


Aufgabe 2.10

Schreibe ein möglichst kurzes Programm für die Melodie:

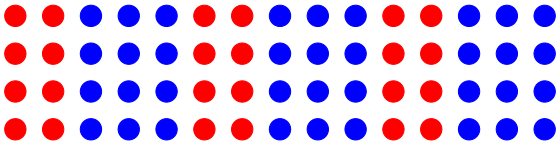


Zur Erinnerung die Frequenzen der Töne:



Aufgabe 2.11

Mit dem Befehl `dot(d)` kann man einen ausgefüllten Kreis mit Durchmesser `d` zeichnen. Schreibe ein Programm, welches das Muster für `d`=15 zeichnet. Der Abstand der Kreise beträgt 25.



Aufgabe 2.12

Was wird mit den beiden Programmen gezeichnet? Skizziere die Ausgaben.

```
1  from gturtle import *
2
3  makeTurtle()
4
5
6  repeat 4:
7      repeat 4:
8          fd(100)
9          rt(90)
10     rt(90)
```

(a)

```
1  from gturtle import *
2
3  makeTurtle()
4  ht()
5
6  repeat 3:
7      rt(30)
8      repeat 3:
9          fd(100)
10         rt(120)
11     rt(90)
```

(b)