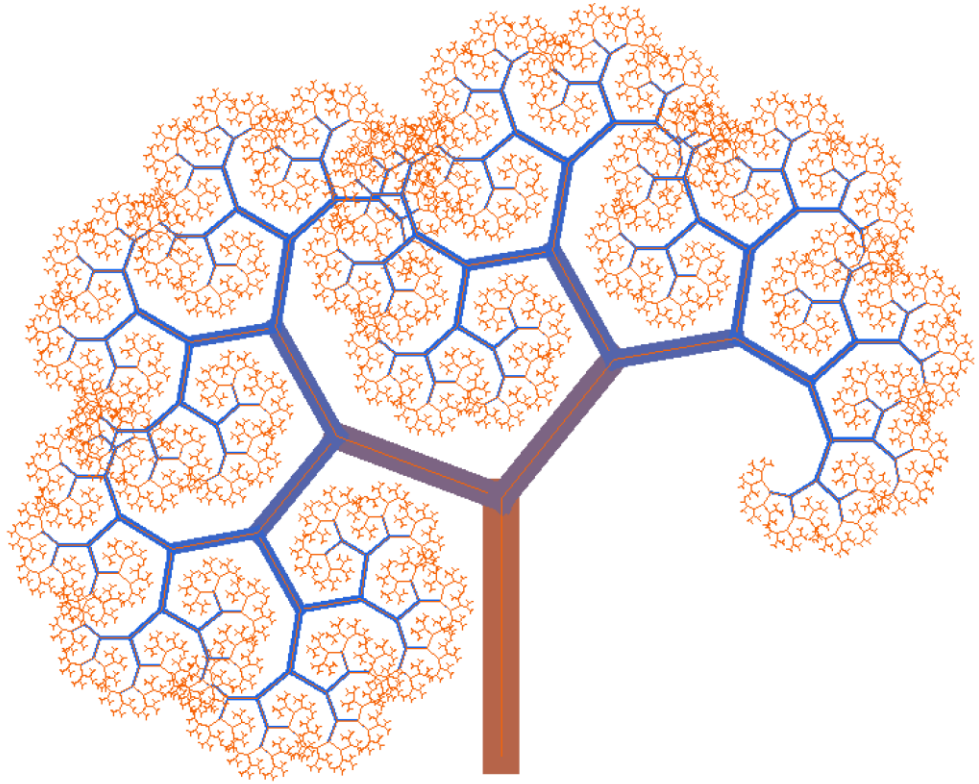
Programmieren mit TigerJython



ABZ
AUSBILDUNGS- UND BERATUNGSZENTRUM
FÜR INFORMATIKUNTERRICHT

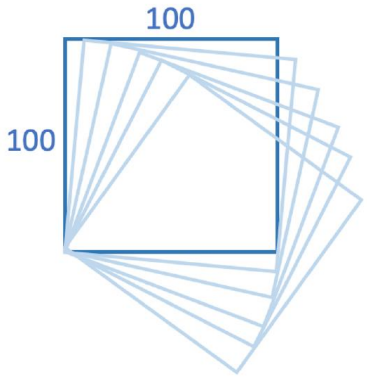
ETH zürich

Einstiegsübung 4

Du hast im Alltag schon viele Animationen kennen gelernt. Sei es in der Form von aufwändigen Animationsfilmen oder auch nur kleine animierte Gif-Bilder.

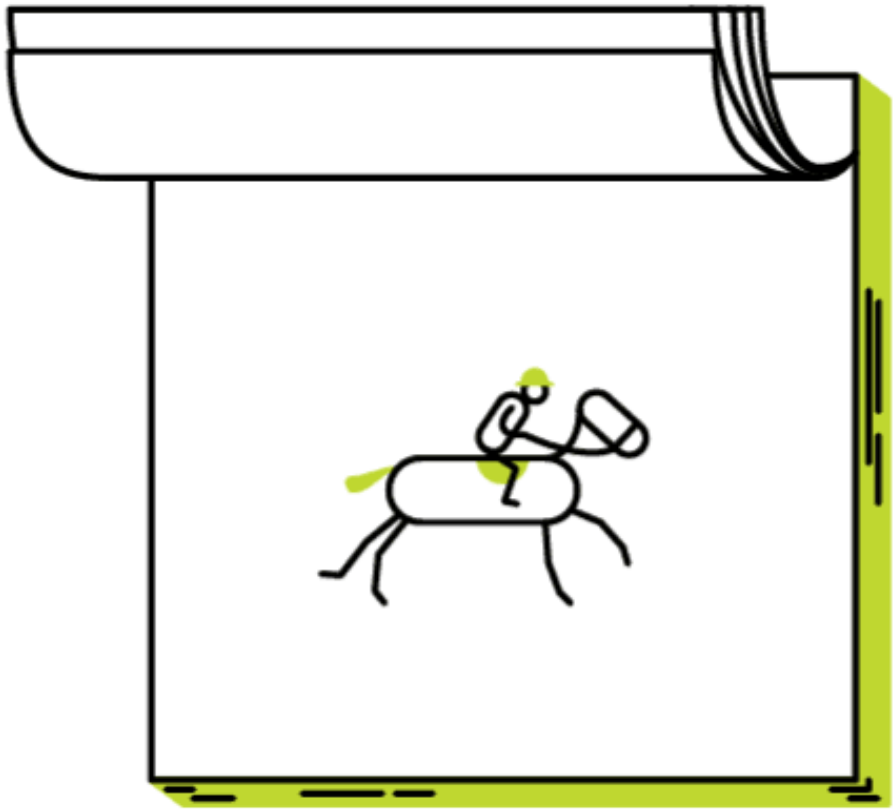
Überlegt euch in einer 2er-Gruppe:

- Welche Grundidee oder Technik steckt hinter der Herstellung von Animationen?
- Wie könnte man so etwas programmieren?



Einstiegsübung 4

Prinzip Daumenkino:



Einstiegsübung 4

Die folgenden vier Schritte werden beliebig oft wiederholt:

1. Ein Bild zeichnen.
2. Kurz warten, damit man es erkennen kann.
3. Das Bild löschen (Alternative: mit der Hintergrundfarbe übermalen).
4. Die Position/Ausrichtung des Bildes ändern.

4. Animationen

Lernziele:

Theorie

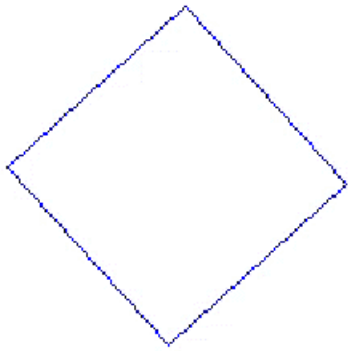
- Du lernst, wie man Animationen programmiert und wie der Bildbuffer zum Speichern und Löschen verwendet werden kann.

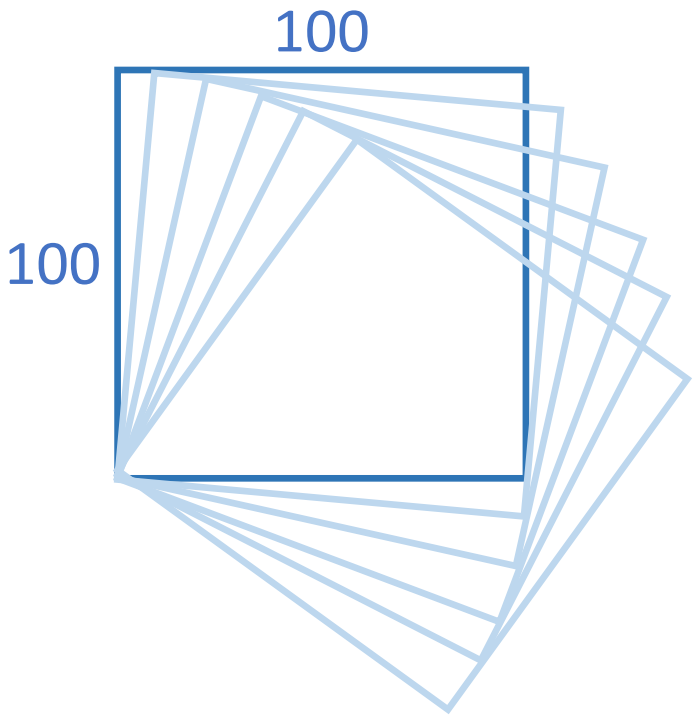
Praxis

- Du wendest gelernte Konzepte wie Schleifen, Befehle und Parameter an und vertiefst sie.

Beispiel 4.1

Wir möchten eine Animation erstellen, bei der sich ein Quadrat um die linke untere Ecke dreht.

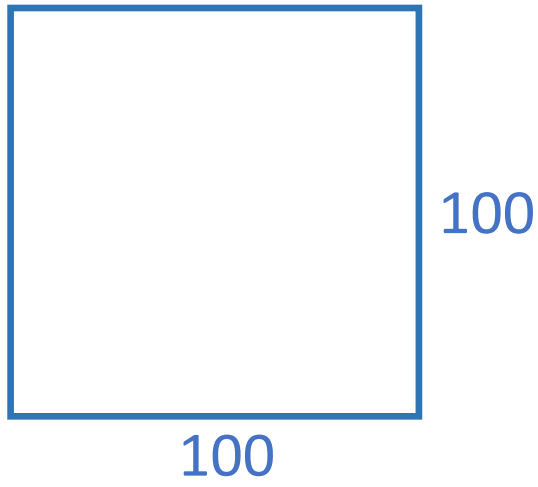




Beispiel 4.1

Wir erstellen dafür zuerst einen Befehl für das Quadrat:

```
1  from gturtle import*
2
3  def quadrat():
4      repeat 4:
5          fd(100)
6          rt(90)
7
```



Beispiel 4.1

Für die Programmierung der Animation erinnern wir uns an den folgenden Ablauf:

→ Wiederholung der folgenden Schritte (beliebig oft):

Schritt 1: Das Quadrat zeichnen

Schritt 2: Kurz warten

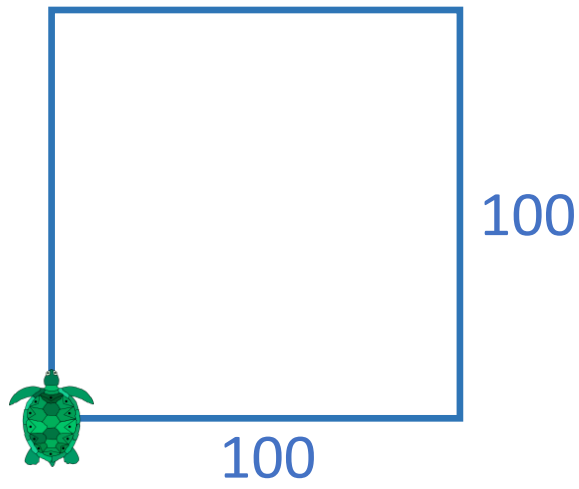
Schritt 3: Das Bild (Quadrat) löschen

Schritt 4: Die Ausrichtung der Turtle ändern.

Beispiel 4.1

Wiederholung der folgenden Schritte (beliebig oft):

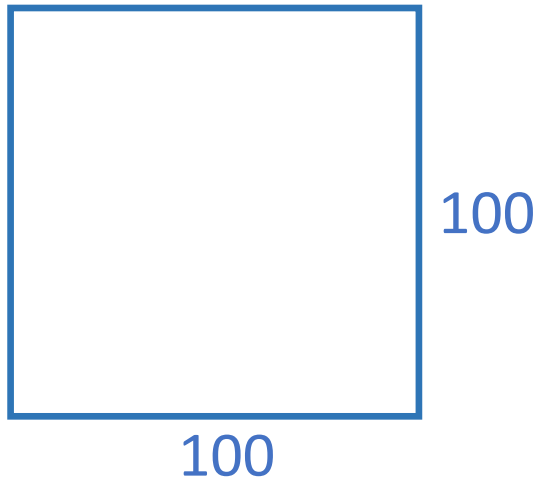
```
1  from gturtle import*
2
3  def quadrat():
4      repeat 4:
5          fd(100)
6          rt(90)
7
8  ##################
9  makeTurtle()
10 ht()
11
12 repeat:
```



Beispiel 4.1

Schritt 1: Zeichnen des Quadrats

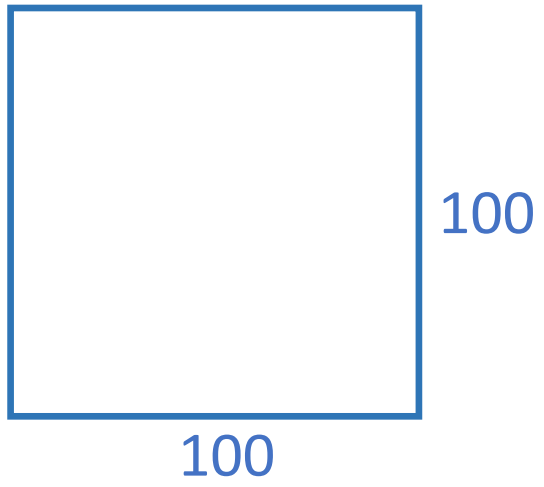
```
1  from gturtle import*
2
3  def quadrat():
4      repeat 4:
5          fd(100)
6          rt(90)
7
8  ##################
9  makeTurtle()
10 ht()
11
12 repeat:
13     quadrat()
```



Beispiel 4.1

Schritt 2: Kurz warten

```
1  from gturtle import*
2
3  def quadrat():
4      repeat 4:
5          fd(100)
6          rt(90)
7
8  ##################
9  makeTurtle()
10 ht()
11
12 repeat:
13     quadrat()
14     delay(50)
```



Wartet 50 Millisekunden bis Wechsel zur nächsten Zeile

Beispiel 4.1

Schritt 3: Grafik löschen

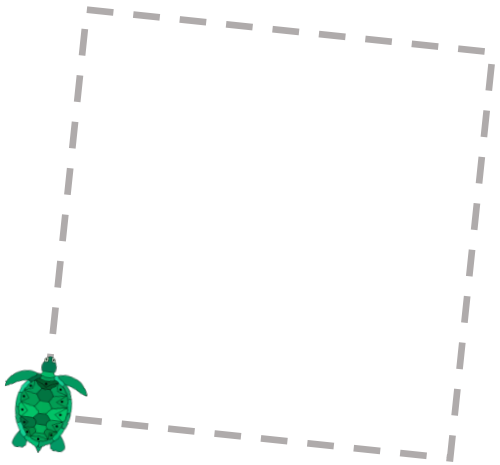
```
1  from gturtle import*
2
3  def quadrat():
4      repeat 4:
5          fd(100)
6          rt(90)
7
8  ##################
9  makeTurtle()
10 ht()
11
12 repeat:
13     quadrat()
14     delay(50)
15     clear()
```

Löscht alles, was bisher gezeichnet wurde

Beispiel 4.1

Schritt 4: Ausrichtung der Turtle ändern

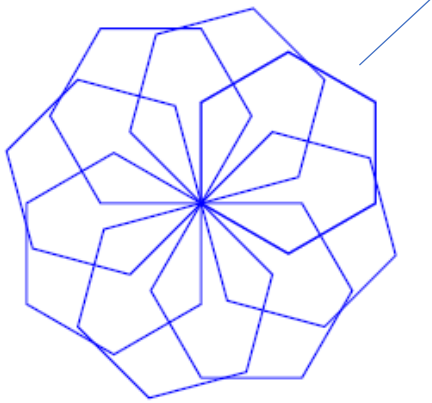
```
1  from gturtle import*
2
3  def quadrat():
4      repeat 4:
5          fd(100)
6          rt(90)
7
8  ##################
9  makeTurtle()
10 ht()
11
12 repeat:
13     quadrat()
14     delay(50)
15     clear()
16     rt(6)
```



Übungsphase

Bearbeite die Aufgaben 4.1 bis 4.3 am Computer

Aufgabe 4.1

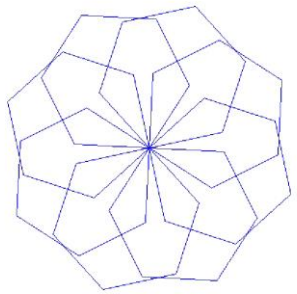


```
from gturtle import*

def sechseck():
    repeat 6:
        fd(100)
        rt(60)

def mandala():
    repeat 8:
        sechseck()
        rt(360/8)

################
```

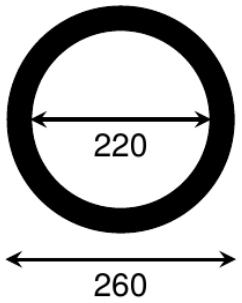
Aufgabe 4.1

```
1  from gturtle import*
2
3  def sechseck():
4      repeat 6:
5          fd(100)
6          rt(60)
7
8  def mandala():
9      repeat 8:
10         sechseck()
11         rt(360/8)
12
13 ################
```

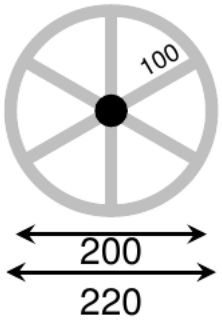
```
makeTurtle()
ht()

repeat:
    mandala()
    delay(50)
    clear()
    rt(6)
```



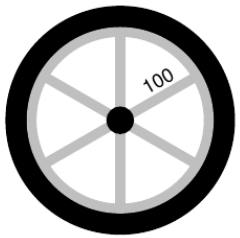
Aufgabe 4.3




```
from gturtle import*

def reifen():
    setPenColor("black")
    dot(260)
    setPenColor("white")
    dot(220)
```




```
def felgen():
    setPenColor("gray")
    dot(220)
    setPenColor("white")
    dot(200)
    setPenWidth(10)
    setPenColor("gray")
    repeat 6:
        fd(100)
        bk(100)
        rt(60)
    setPenColor("black")
    dot(30)
```

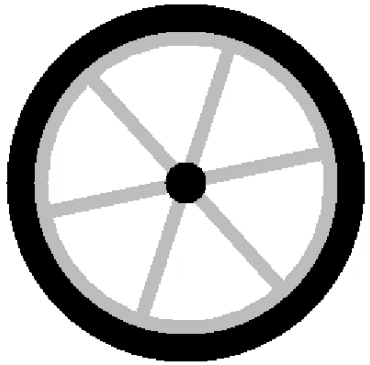



```
def rad():
    reifen()
    felgen()
```

Aufgabe 4.3

```
from gturtle import*

def reifen():
    setPenColor("black")
    dot(260)
    setPenColor("white")
    dot(220)

def felgen():
    setPenColor("gray")
    dot(220)
    setPenColor("white")
    dot(200)
    setPenWidth(10)
    setPenColor("gray")
    repeat 6:
        fd(100)
        bk(100)
        rt(60)
    setPenColor("black")
    dot(30)

def rad():
    reifen()
    felgen()
####################3
```

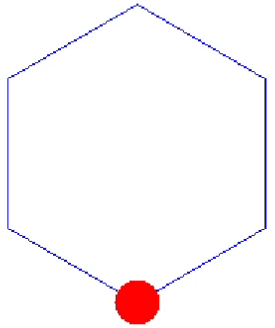
```
26 ##########################

27

28 makeTurtle()
29 ht()
30
31 repeat:
32     rad()
33     delay(25)
34     clear()
35     rt(3)
```



c) 1 Umdrehung in 3 s
→ 360° in 3000 ms
→ 6° in 50 ms (3° in 25 ms)

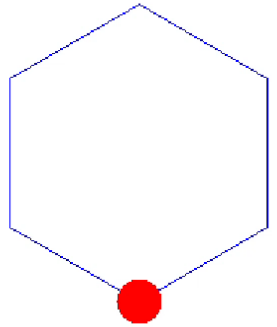
Beispiel 4.2

In diesem Beispiel soll ein roter Kreis von Ecke zu Ecke eines regelmässigen Sechsecks „springen“ und das Sechseck an Ort und Stelle bleiben:



Beispiel 4.2

```
1  from gturtle import*
2
3  def sechseck():
4      repeat 6:
5          fd(100)
6          rt(60)
7
8  def Kreis():
9      setPenColor("red")
10     dot(30)
11 ########################
12 makeTurtle()
13 ht()
14
15 sechseck()
16 savePlayground()
17
18 repeat:
19     Kreis()
20     delay(1000)
21     clear()
22     pu()
23     fd(100)
24     rt(60)
25     pd()
```



`savePlaygound()` speichert alle, vor dem Aufruf dieses Befehls, programmierten Grafiken in einem Bildbuffer.
Hier also das Sechseck.

Dieses wird mit `clear()` nicht gelöscht!

Ausblick -Variablen