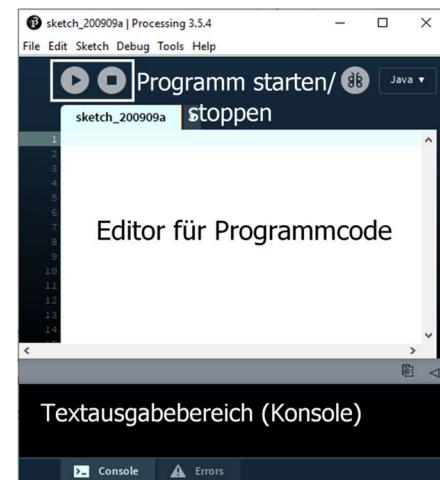


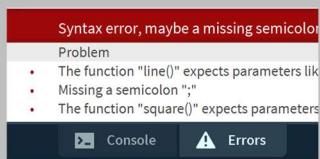
Processing ist ein Java-Dialekt und besonders für Programmieranfänger geeignet. Ein Programm wird in Processing auch **Sketch** genannt. Anders als in Scratch wird der Programmcode von einem Sketch **als Text** geschrieben. Dabei muss man **genaue Regeln („Syntax“)** beachten, sonst kann das Programm nicht ausgeführt werden.

Beim Ausführen des Programms wandelt Java den Programmcode in eine ausführbare Datei um und startet das Programm. In Processing erzeugt jeder Sketch ein **Programmfenster** in dem Zeichnungen erstellt oder Texte angezeigt werden können. Zusätzlich kann ein Programm auch Texte in der **Konsole** anzeigen.

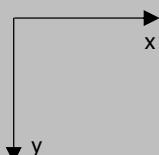
Hier kann man Processing kostenlos herunterladen:
<https://processing.org/download/>



Bei einem *Syntaxfehler* erscheint beim Ausführen ein Fehler im unteren Programmbericht. Dann kann dein ganzes Programm nicht gestartet werden:



Die y-Werte werden in Processing anders als in Mathe angegeben („nach unten“).



1 Öffne Processing und tippe die Anweisungen ab. Beschreibe, was die jeweiligen Anweisungen bewirken. Verändere die Zahlen in den Anweisungen. Erkläre die Bedeutung der Zahlen.

a) `line(10, 20, 50, 90);`

Lösche nach jeder Teilaufgabe die alte Anweisung.

b) `rect(5, 10, 70, 20);`

c) `square(20, 50, 40);`

d) `circle(50, 30, 60);`

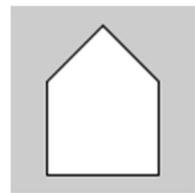
2 a) Erstelle diesen Sketch:

```
ellipse(40, 40, 35, 35);
rect(40, 40, 40, 30);
triangle(60, 60, 20, 90, 60, 90);
```

Verändere die Reihenfolge der Anweisungen. Was fällt dir auf?

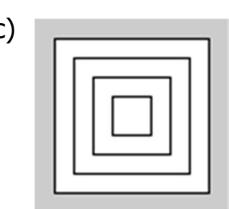
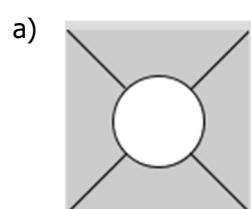
b) Bearbeite die Reihenfolge der Anweisungen, dass das Haus gezeichnet wird.

```
beginShape();
vertex(20, 40);
vertex(80, 90);
vertex(80, 40);
vertex(20, 90);
vertex(20, 90);
vertex(50, 10);
endShape();
```



3 Erstelle einen Sketch um die Figuren zu zeichnen.

Speicher jeden Sketch ab (Dateiname z.B. Kap1A3a).

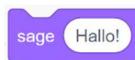


Anweisungen in Java

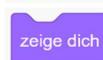
Aus Scratch-Blöcken werden Anweisungen.

Unsere ersten **Anweisungen** sind Befehle, die Processing bereits kennt und uns zur Verfügung stellt. Die weiteren Angaben in der runden Klammer nennt man **Parameter**. Jede Anweisung wird mit einem Semikolon ; beendet.

Bsp.:



wird zu: `sage ("Hallo");`



wird zu: `zeigedich();`

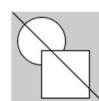


wird zu: `gehezu(0, 0);`

Die Anweisungen `sage`, `gehezu` und `zeigedich` gibt es in Processing nicht. Sie sollen dir nur die Schreibweise (Syntax) zeigen.

Schreibt man mehrere Anweisungen untereinander entsteht eine **Sequenz**. Das Programm arbeitet alle Anweisungen in einer Sequenz **nacheinander von oben nach unten** ab.

Bsp.: `circle(30, 30, 50);`
`square(30, 40, 50);`
`line(0, 0, 100, 100);`



`line(0, 0, 100, 100);`
`square(30, 40, 50);`
`circle(30, 30, 50);`

Zeichnungen verdecken vorherige Zeichnungen!

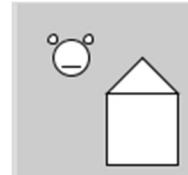


Tipp:

Schreibe jede Anweisung in eine neue Zeile. Mit Leerzeilen kannst du dein Programm in Teile gruppieren, um den Programmcode übersichtlicher zu gestalten.

`circle(30,30,20);`
`circle(20,20,5);`
`circle(40,20,5);`
`line(25,35,35,35);` **Übersichtlich!**

`rect(50,50,40,40);`
`triangle(50,50,70,30,90,50);`



`circle(30,30,20); circle(20,20,5);`
`line(25,35,35,35);`
`rect(50,50,40,40);`
`triangle(50,50,70,30,90,50);`

Chaos...

Codierung von Farben:
Computer codieren Farben meistens mit dem RGB-System. Dabei gibt man drei Zahlenwerte für die Farben Rot, Grün und Blau an. Die Werte reichen von 0 bis 255.

Beispiel:

Rot = 255, 0, 0
Weiß = 255, 255, 255
Gelb = 255, 255, 0
Schwarz = 0, 0, 0
Grau = 50, 50, 50

Alpha-Kanal:

Möchte man mit neuen Zeichnungen die alten nicht vollständig überdecken, kann man mit einer vierten Zahl angeben, wie transparent die neue Zeichnung sein soll. Dies nennt man oft den „Alpha-Wert“.

Beispiel:

```
fill(255,0,0);
rect(0,50,100,50);
fill(0,255,0,100);
circle(50,50,50);
```

Tipp für die Farben bei 5b:

Rechteck links: 100, 100, 255
Rechteck rechts: 0, 255, 0
Innere Farbe: 0, 255, 200
oder mit Graustufen:
Rechteck links: 255
Rechteck rechts: 0
Innere Farbe: 150

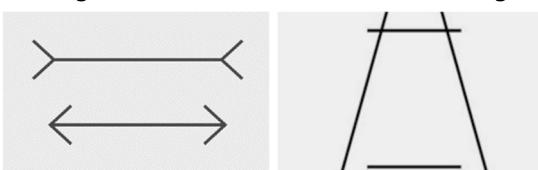
4 ③ Es gibt Anweisungen, die das Verhalten von späteren Zeichenanweisungen verändern. Probiere die Beispiele aus, um herauszufinden, was die Anweisungen `stroke`, `strokeWeight`, `noStroke`, `fill` und `noFill` bewirken.

- a) `stroke(255, 0, 0);`
`circle(50, 50, 80);`
- b) `fill(0, 0, 255);`
`circle(50, 50, 80);`
- c) `strokeWeight(5);`
`circle(50, 50, 80);`

- d) `noStroke();`
`circle(50, 50, 80);`
- e) `noFill();`
`circle(50, 50, 80);`
- f) Probiere Kombinationen aus den Anweisungen aus und zeichne auch mehrere Objekte.

5 ③ Optische Täuschungen: Erstelle für jede optische Täuschung einen Sketch. Verwende die Vorlage rechts.

a)



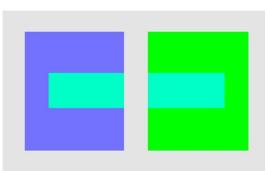
Die horizontalen Linien sind gleich lange, erscheinen aber verschieden lange.

Sketch Kap1_A5

```
size(600, 400);
background(200);
```

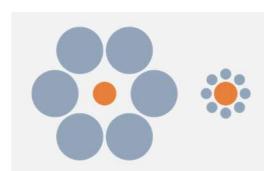
Notiere ab hier deine weiteren Anweisungen. Du findest sicher heraus, was die size- und background-Anweisungen bewirken.

b)



Die inneren Rechtecke haben dieselbe Farbe, erschienen aber verschieden.

c)



Die inneren Kreise sind gleich groß, erschienen aber verschieden groß.