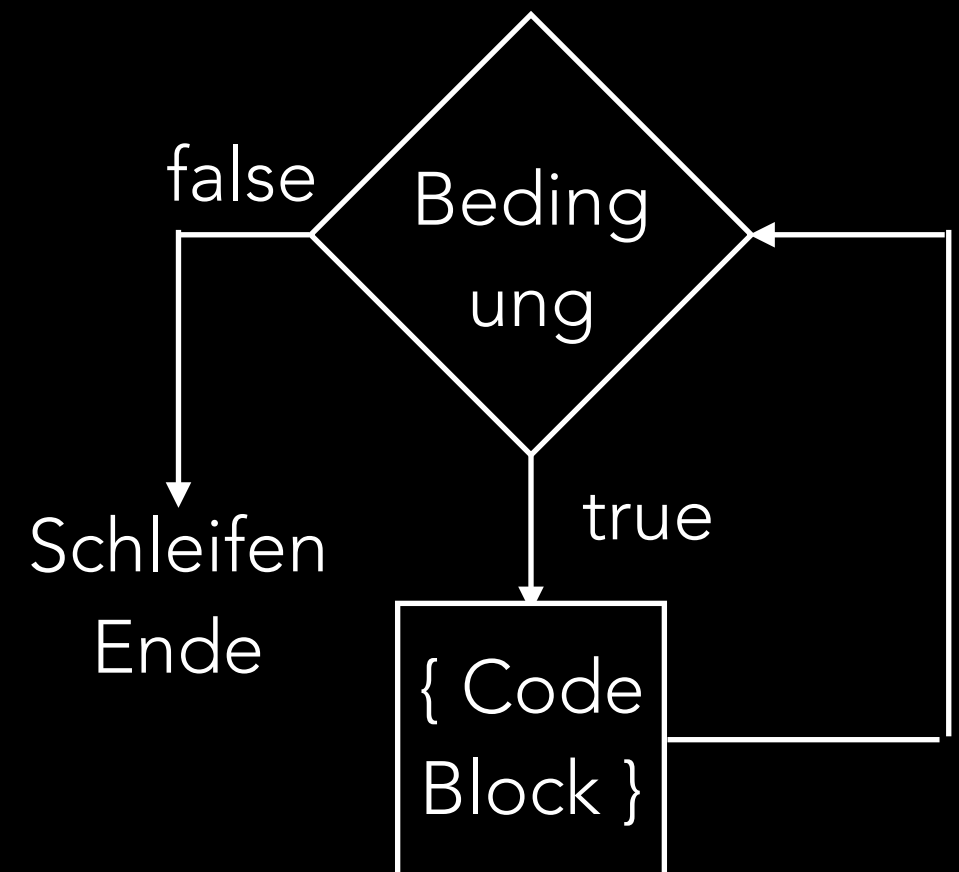


BRÜCKENKURS PROGRAMMIEREN - FIONA NÜESCH

# REKAPITULATION

# Die **while**-Schleife

```
boolean abbruchsbedingung;  
  
while ( abbruchsbedingung ){  
    // Anweisungs Block  
}  
  
do {  
    // Anweisungs Block  
} while ( abbruchsbedingung );
```



```
int max = 10;  
int sum = 10;  
  
while ( sum < max ){  
    sum = sum + 2;  
}  
println(sum); 10
```

```
sum = 10;  
  
do{  
    sum = sum + 2;  
}while ( sum < max );  
println(sum); 12
```

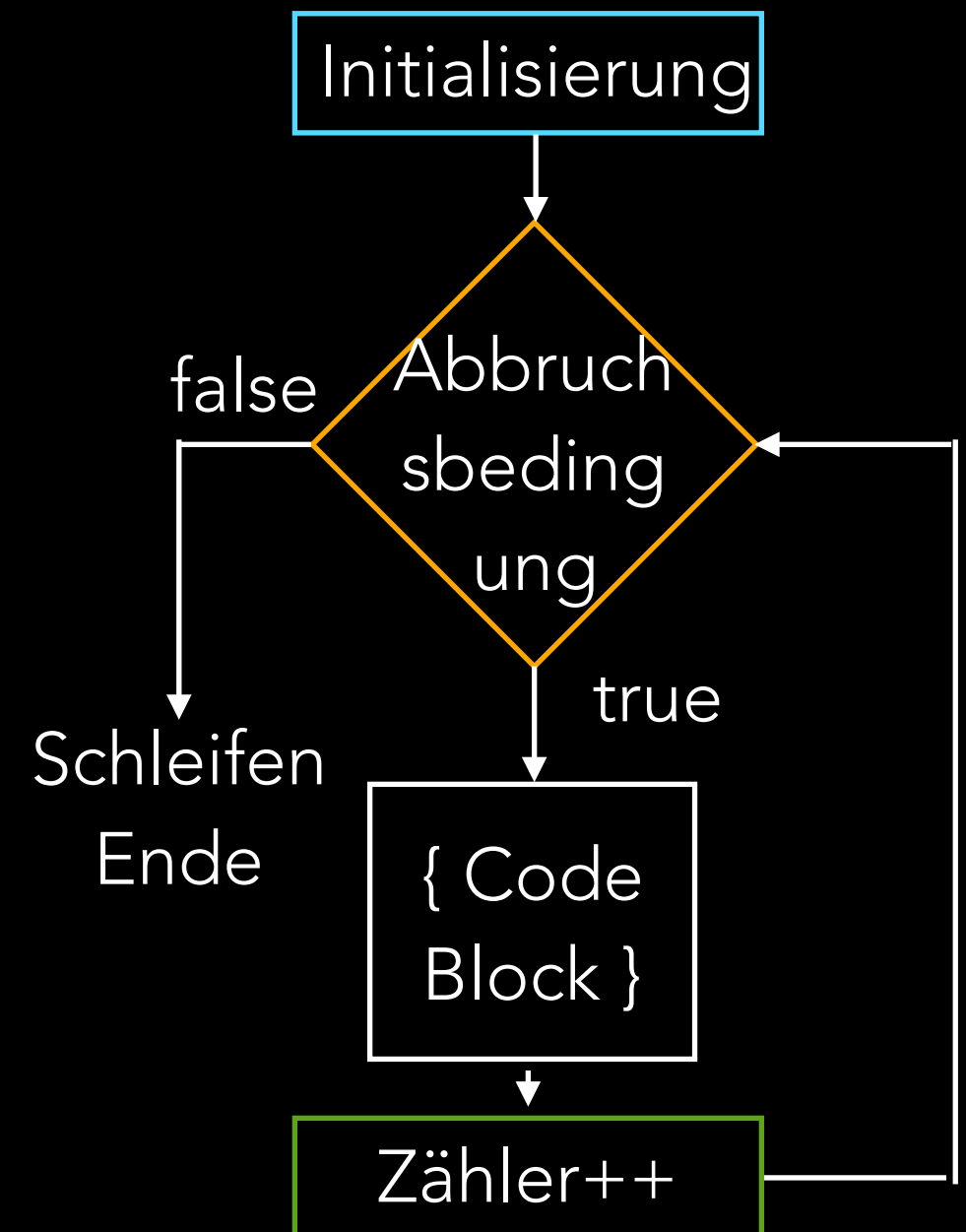
# Die **for**-Schleife

```
int max = 10;  
  
for( int i = 0; i < max; i++){  
    // Anweisungs Block  
}
```

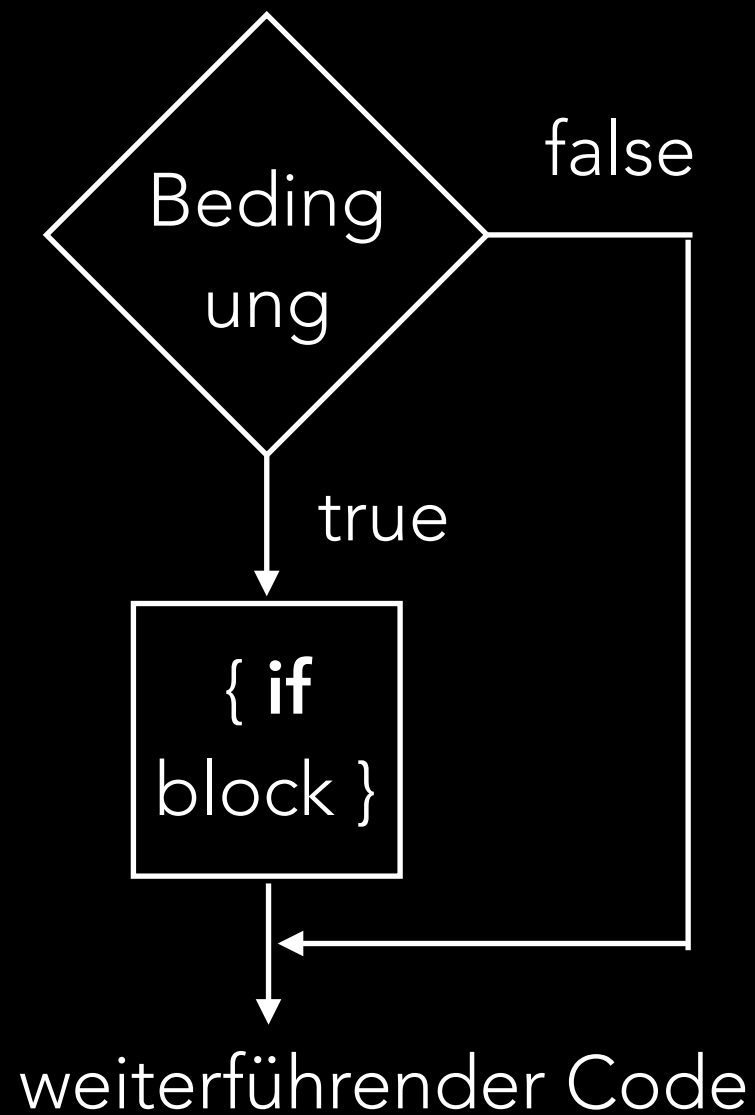
Update der  
Zähler Variable

Abbruchsbedingung

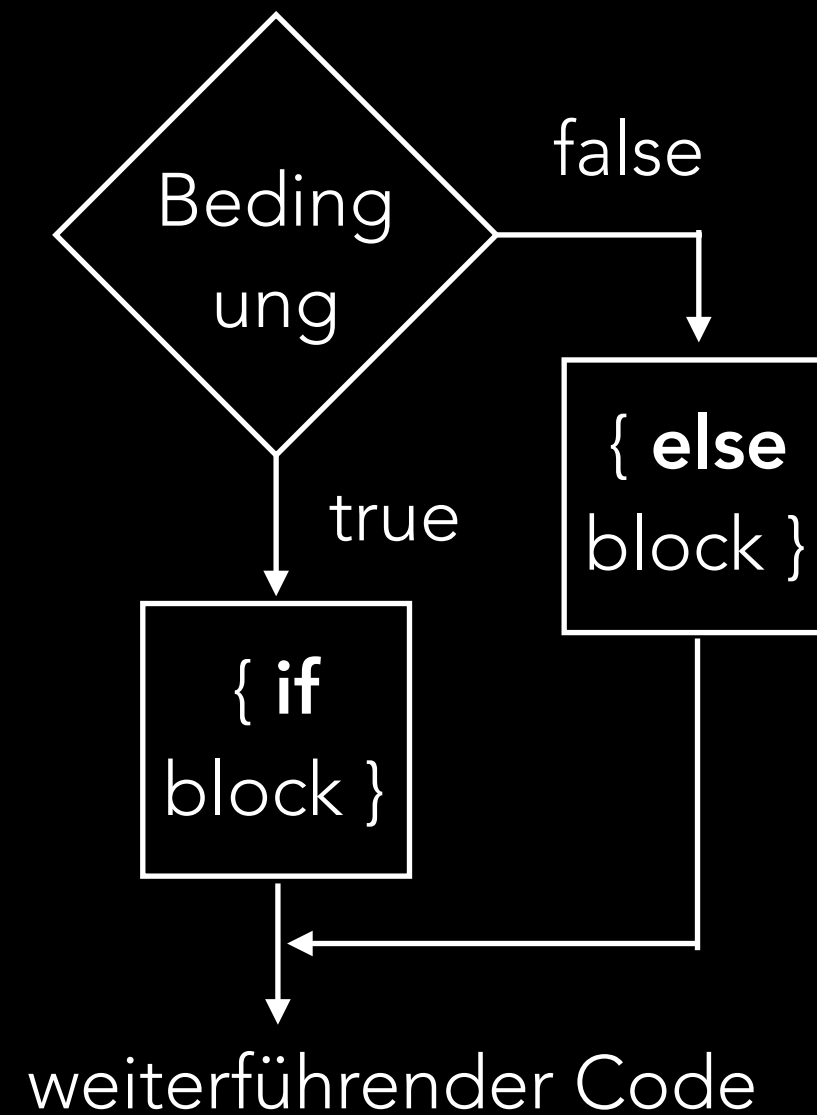
Initialisierung der Zähler  
Variable vom Typ int



```
if( a < b ){  
    // if code block  
}
```



```
if( a < b ){  
    // if code block  
}else{  
    // else code block  
}
```



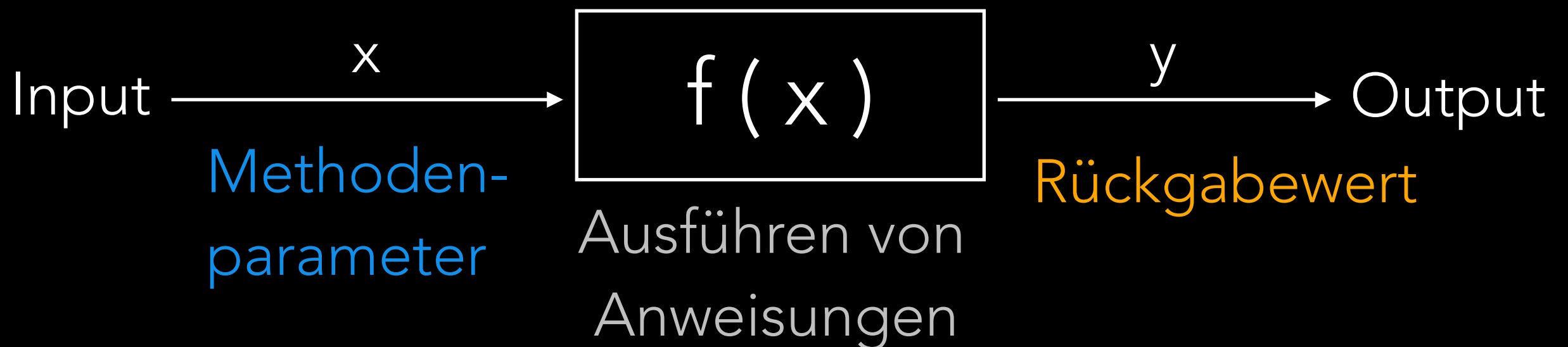
Eine **Methode** kann sich (in etwa) wie eine mathematische Funktion vorgestellt werden

Funktionsargument

$$y = f(x)$$

Resultat

Ausführen von Berechnung



Rückgabe Typ  
( Datentyp des Outputs )

Methodenname

Methodenparameter  
( Input mit Typ und Name )

return Statement

- Definiert welche Variable zurückgegeben wird.
- Muss dem Typ des Rückgabewerts entsprechen.
- Beendet die Methode.

```
int line(int x){  
    int y = 2 * x + 1;  
    return y;  
}
```

Methoden  
Code  
Block

```
int result = line(2);
```

Methoden Aufruf



# CAST OPERATION

immer möglich:

```
int i = 1;  
float f = i;
```

nicht möglich:

```
7 float f = 1.0;  
8 int i = f;
```

Type mismatch, "float" does not match with "int"

Ein Cast ermöglicht es einen grösseren Datentypen in einen kleineren zu verwandeln. Wir als Programmierende nehmen den möglichen Verlust in Kauf.

```
float f = 4.0;  
int i = (int) f;
```

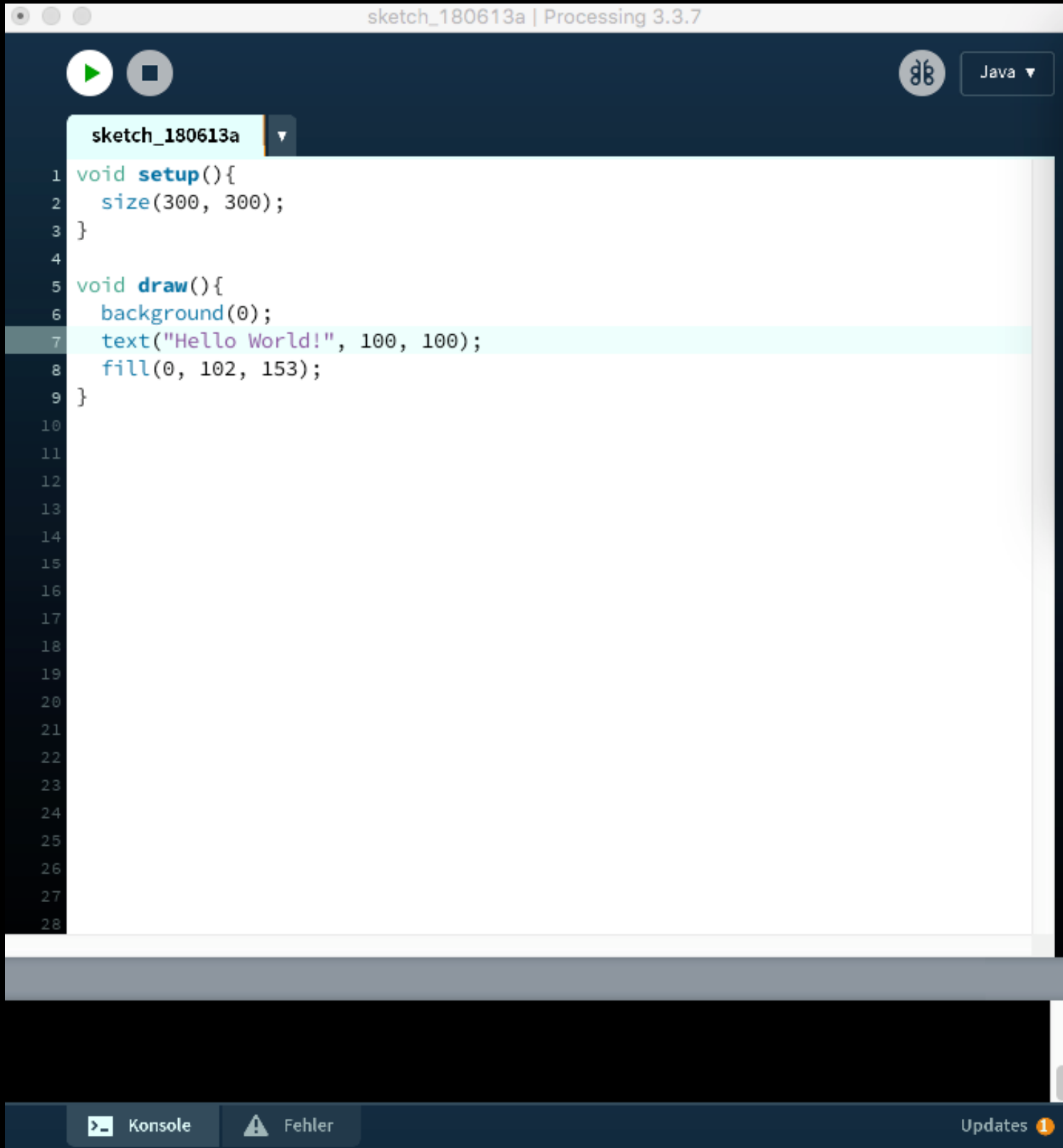
```
float fl = 4.89;  
int in = (int) fl;
```

Cast

i und in sind 4 (wird einfach abgeschnitten, nicht gerundet)

PROCESSING





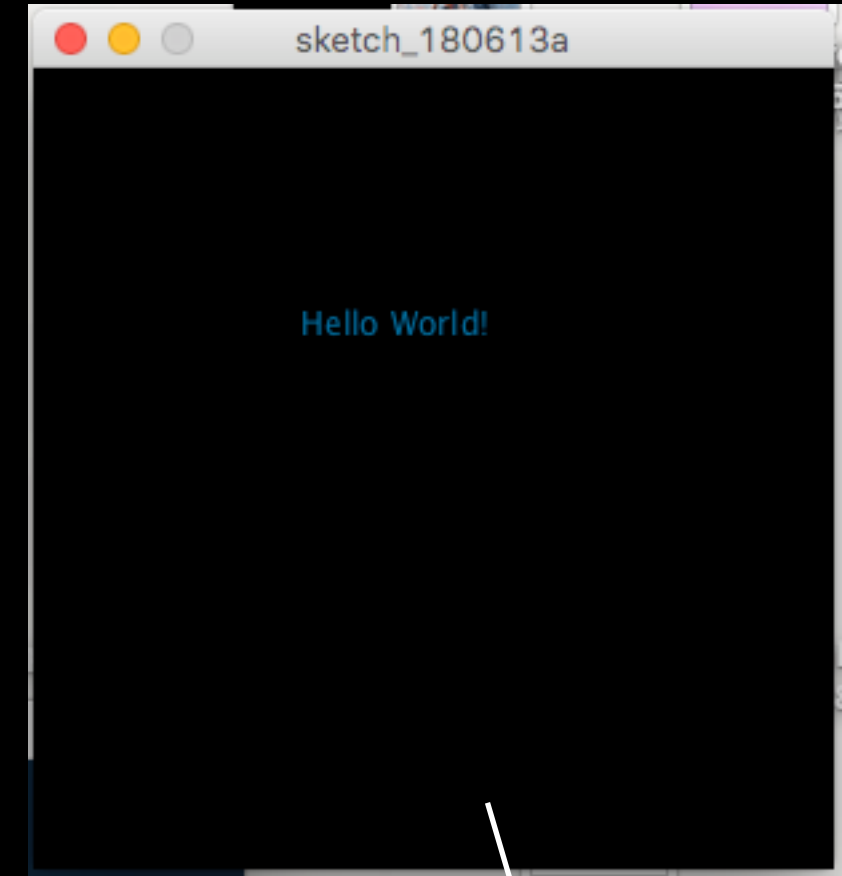
The image shows the Processing IDE interface. The title bar reads "sketch\_180613a | Processing 3.3.7". The code editor contains the following code:

```
1 void setup(){
2   size(300, 300);
3 }
4
5 void draw(){
6   background(0);
7   text("Hello World!", 100, 100);
8   fill(0, 102, 153);
9 }
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

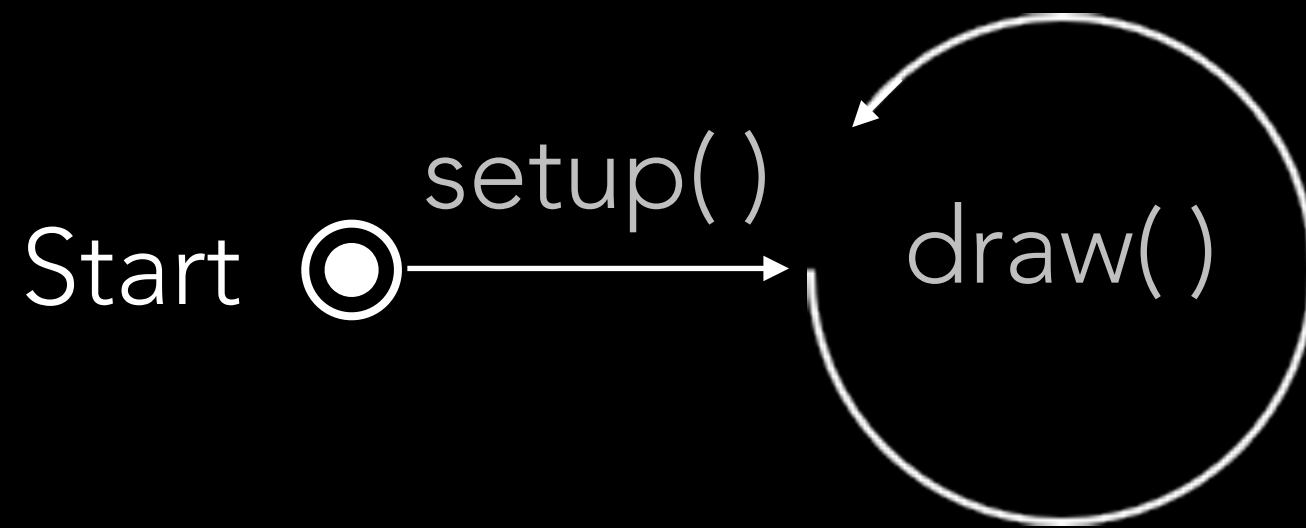
The code is written in a dark-themed editor with a light blue background. The line numbers are on the left. The code is as follows:

```
1 void setup(){
2   size(300, 300);
3 }
4
5 void draw(){
6   background(0);
7   text("Hello World!", 100, 100);
8   fill(0, 102, 153);
9 }
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

The IDE has a toolbar at the top with a play button, a stop button, and a language dropdown menu set to "Java". At the bottom, there are tabs for "Konsole" and "Fehler", and an "Updates" button with a notification icon.



Display Window

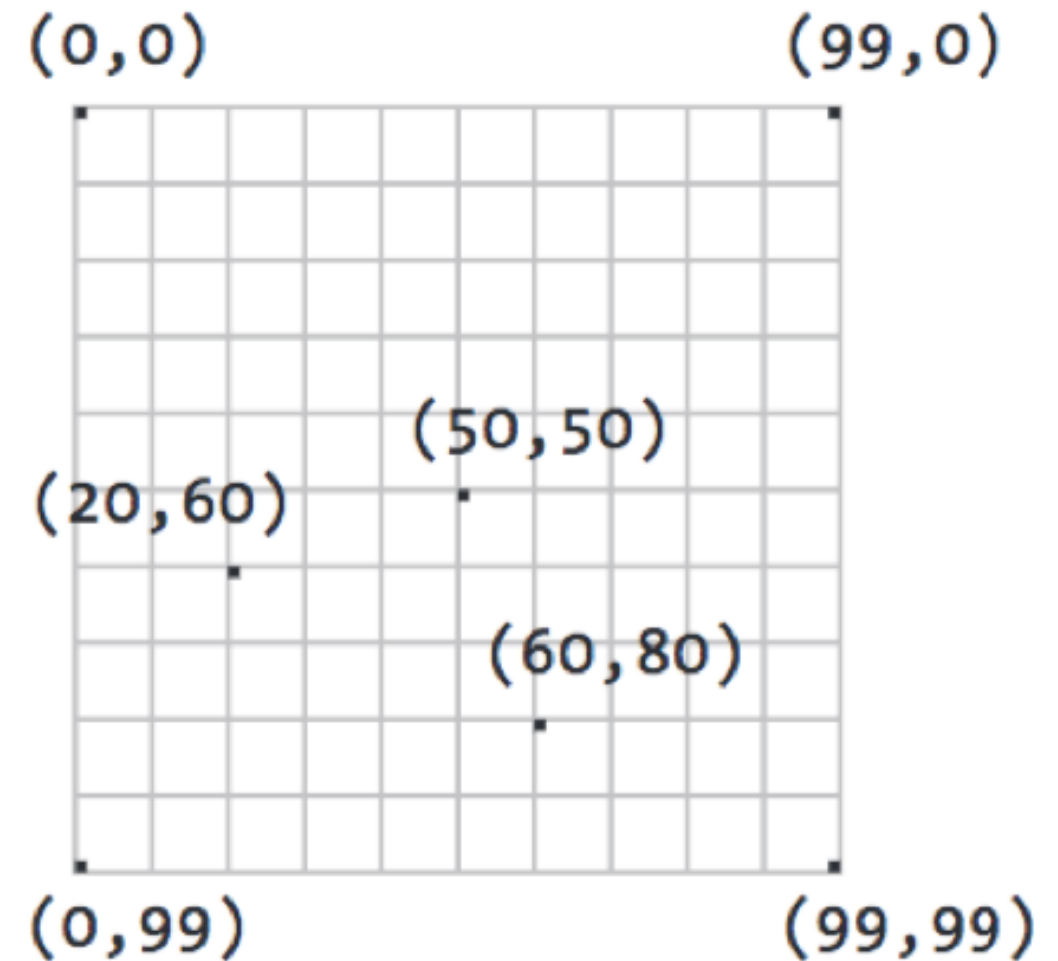
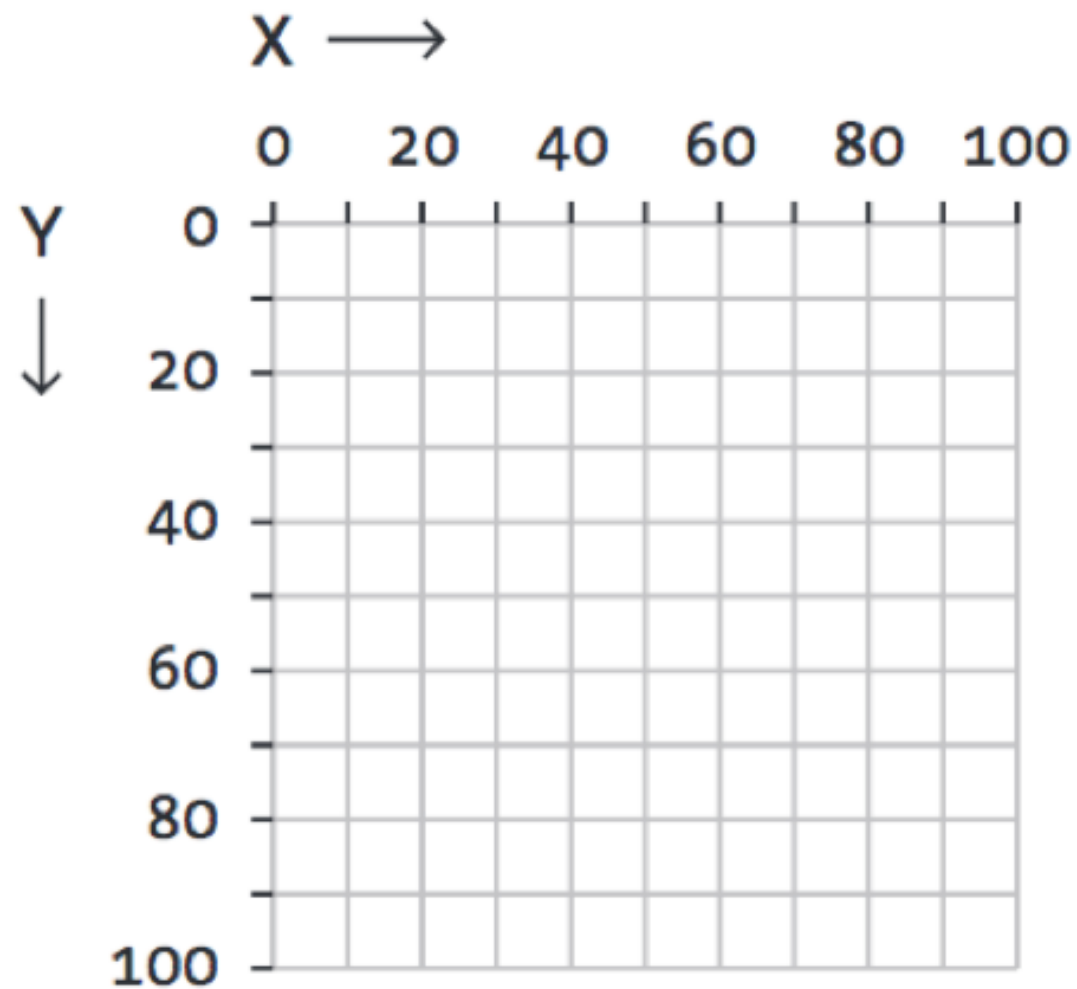


sketch\_180613a | Processing 3.3.7

sketch\_180613a ▼

```
1 void setup(){
2   size(300, 300);
3 }
4
5 void draw(){
6   background(0);
7   text("Hello World!", 100, 100);
8   fill(0, 102, 153);
9 }
10
11
```

# Das Koordinatensystem



# Erste Anweisungen

## **size( int: width, int: height )**

Definiert die Dimension des display window. Entspricht Pixeln ( bzw. der Breite und Höhe des Koordinatensystems ). Muss immer als erstes und in der setup() Methode definiert werden.

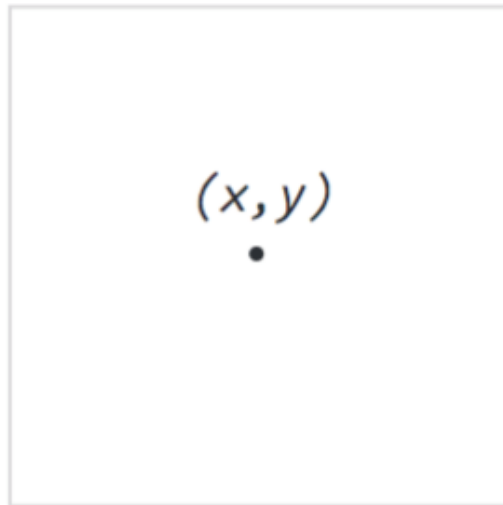
[https://processing.org/reference/size\\_.html](https://processing.org/reference/size_.html)

## **background( color: color )**

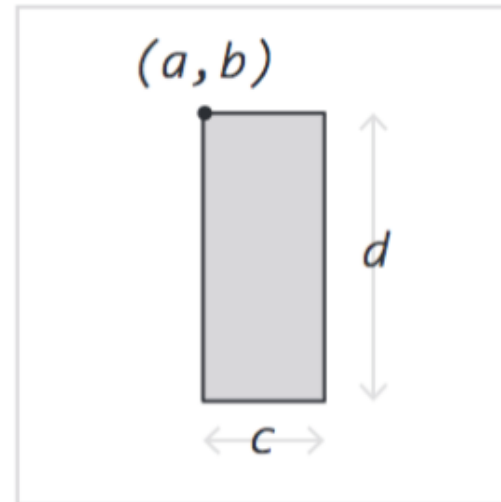
Setzt die Hintergrundfarbe im display Window.

[https://processing.org/reference/background\\_.html](https://processing.org/reference/background_.html)

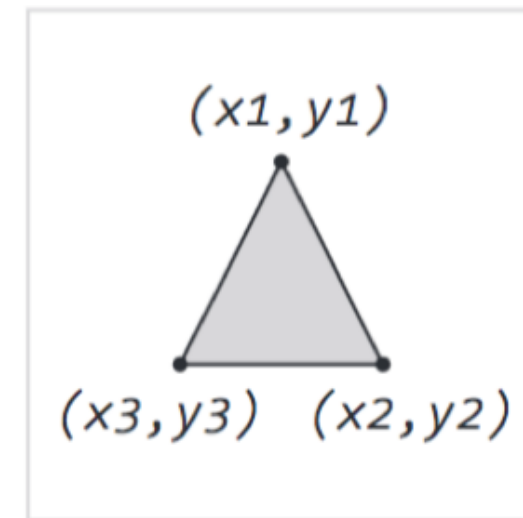
# Erste Anweisungen



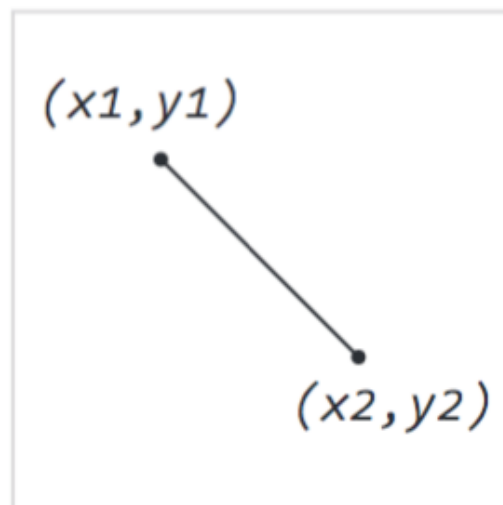
`point(x, y)`



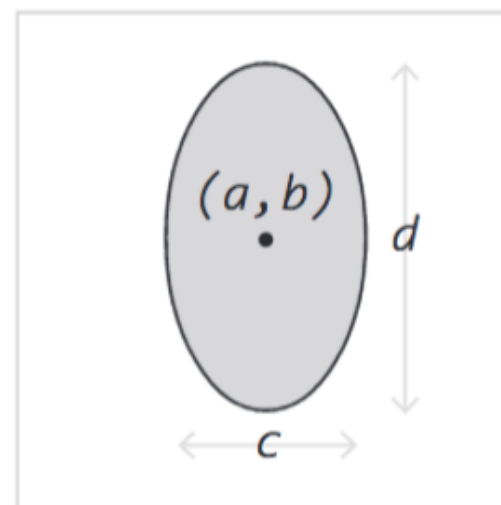
`rect(a, b, c, d)`



`triangle(x1, y1, x2, y2, x3, y3)`



`line(x1, y1, x2, y2)`



`ellipse(a, b, c, d)`

# Erste Anweisungen

**color colorName = color( int: v1, int: v2, int: v3 )**

Erzeugt eine Farbe. **color()** ist die Methode um eine Farbe zu erzeugen. **color** ist ein neuer Datentyp von Processing, der eine Farbe speichern kann

[https://processing.org/reference/color\\_.html](https://processing.org/reference/color_.html)

**fill( color: rgb )**

Setzt die Farbe mit der die nachfolgenden Formen gefüllt werden sollen.

[https://processing.org/reference/fill\\_.html](https://processing.org/reference/fill_.html)

# Erste Anweisungen

## **stroke( color: rgb )**

Setzt die Farbe für den Rahmen der nachfolgenden Formen.

[https://processing.org/reference/stroke\\_.html](https://processing.org/reference/stroke_.html)

## **noStroke( )**

Deaktiviert den Rahmen für die nachfolgenden Formen.

[https://processing.org/reference/noStroke\\_.html](https://processing.org/reference/noStroke_.html)

# Erste Anweisungen

**random( float: max ), random( float: min, float: max )**

Gibt einen zufälligen Wert aus. Entweder von 0 bis Max. Oder von Min bis Max.

[https://processing.org/reference/random\\_.html](https://processing.org/reference/random_.html)



Dokumentation: <https://processing.org/reference/>

[Processing](#)[p5.js](#)[Processing.py](#)[Processing for Android](#)[Processing for Pi](#)[Processing Foundation](#)

# Processing

[Cover](#)[Download](#)[Donate](#)[Exhibition](#)[Reference](#)[Libraries](#)[Tools](#)[Environment](#)[Tutorials](#)[Examples](#)[Books](#)[Handbook](#)[Overview](#)[People](#)[Shop](#)

Reference. Processing was designed to be a flexible software sketchbook.

## Structure

`()` (parentheses)  
`,` (comma)  
`.` (dot)  
`/**/` (multiline comment)  
`/** */` (doc comment)  
`//` (comment)  
`;` (semicolon)  
`=` (assign)  
`[]` (array access)  
`{ }` (curly braces)  
`catch`  
`class`  
`draw()`  
`exit()`  
`extends`

## Shape

`createShape()`  
`loadShape()`  
`PShape`  
  
2D Primitives  
`arc()`  
`ellipse()`  
`line()`  
`point()`  
`quad()`  
`rect()`  
`triangle()`

## Curves

## Color

### Setting

`background()`  
`clear()`  
`colorMode()`  
`fill()`  
`noFill()`  
`noStroke()`  
`stroke()`

### Creating & Reading

`alpha()`  
`blue()`  
`brightness()`  
`color()`

GLOBALE VARIABLEN

## Scope der Variable i

```
{  
  int i = 1;  
  
  {  
    println(i);  
  }  
}
```

```
println(i);
```

The variable "i" does not exist

## Globaler Scope

```
1  
2 int x;  
3  
4 void setup(){  
5  
6 }  
7  
8 void draw(){  
9  
10 }  
11
```

# Beispiel

```
1  int x;  
2  
3  void setup(){  
4      frameRate(5);  
5      x = 0;  
6  }  
7  
8  void draw(){  
9      increaseX();  
10     println(x);  
11 }  
12  
13  
14 void increaseX(){  
15     x++;  
16 }
```

```
1  
2  
3  
4  
5  
6  
7  
9  
10  
11  
12  
13  
14  
15  
16  
--
```

## Übung Processing - Erste Anweisungen

„Es wird vielleicht nicht einfacher, aber du wirst  
immer besser.“

–INTERNET

USER INPUT

# Processing Variablen

## **width, height**

Variablen die die Breite und Höhe des Display Windows speichern

[https://processing.org/reference/width\\_.html](https://processing.org/reference/width_.html)

## **mouseX, mouseY**

Variablen die die x und y Postion der Mouse speichern.

<https://processing.org/reference/mouseX.html>

## **key, keyCode**

Enthält den Wert der Taste die als letztes gedrückt wurde.

<https://processing.org/reference/mouseX.html>

# User Input Events

## **mousePressed()**

Funktion die aufgerufen wird wenn man mit der Maus klickt.

[https://processing.org/reference/width\\_.html](https://processing.org/reference/width_.html)

## **keyPressed()**

Funktion die aufgerufen wird wenn eine Taste gedrückt wird.

<https://processing.org/reference/mouseX.html>



Übung Processing - User Input

„Zitat hier eingeben.“

–CHRISTIAN BAUER



- Lade pacman.pde herunter
- Öffne den Sketch und füge da einen gelben Kreis hinzu, den man über die Maustasten bewegen kann.
- Füge 10 rote Quadrate hinzu über die man mit dem Kreis nicht drüber fahren kann.
- Zusatz: füge eine Animation hinzu, wenn man mit dem Kreis an ein Rechteck stösst.